Iterative projection algorithms and applications in x-ray crystallography

Victor Lo

Department of Electrical and Computer Engineering

A thesis presented for the degree of
Doctor of Philosophy

University of Canterbury
Christchurch, New Zealand
October 2010

To my loving parents, Catherine and Nian

Contents

Abstract

X-ray crystallography is a technique for determining the structure (positions of atoms in space) of molecules. It is a well developed technique, and is applied routinely to both small inorganic and large organic molecules. However, the determination of the structures of large biological molecules by x-ray crystallography can still be an experimentally and computationally expensive task. The data in an x-ray experiment are the amplitudes of the Fourier transform of the electron density in the crystalline specimen. The structure determination problem in x-ray crystallography is therefore identical to a phase retrieval problem in image reconstruction, for which iterative transform algorithms are a common solution method.

This thesis is concerned with iterative projection algorithms, a generalized and more powerful version of iterative transform algorithms, and their application to macromolecular x-ray crystallography. A detailed study is made of iterative projection algorithms, including their properties, convergence, and implementations. Two applications to macromolecular crystallography are then investigated. The first concerns reconstruction of binary image and the application of iterative projection algorithms to determining molecular envelopes from x-ray solvent contrast variation data. An effective method for determining molecular envelopes is developed. The second concerns the use of symmetry constraints and the application of iterative projection algorithms to *ab initio* determination of macromolecular structures from crystal diffraction data. The algorithm is tested on an icosahedral virus and a protein tetramer. The results indicate that *ab initio* phasing is feasible for structures containing 4-fold or 5-fold non-crystallographic symmetry using these algorithms if an estimate of the molecular envelope is available.

Acknowledgements

I would like to express my gratitude to my supervisor Prof Rick Millane. Without his constant guidance and encouragement, this thesis would not have been possible. My special thanks also go to our collaborator Assoc. Prof. Richard Kingston from the University of Auckland. He has provided valuable information on the current state of the art in crystallography. I am especially indebted to Prof Veit Elser from Cornell University whose discussions helped shape my thoughts on projection algorithms, as well as inspiring many elements of the relevant chapter. I am also grateful to Prof Phil Bones and Dr Philippa Martin for their advice and encouragement.

Many thanks to my colleagues and friends in the Computational Imaging and Power Systems Group for their assistance and companionship, especially David Wojtas, Amanda Zhang, Andrew Lapthorn, Irvin Chew, Lance Frater and Michael Hwang.

A big thank you to my twin sister Victoria for providing constant companionship, even before I was born.

Last but not least, it is my parents who have supported me the most throughout my upbringing and education. To my mother Catherine and father Nian, I cannot thank you enough.

Preface

This research was initiated in an effort to apply recently developed iterative projection algorithms to the problem of image reconstruction, or structure determination, in x-ray crystallography, as well as to gain a greater understanding of the behaviour of projection algorithms themselves.

The thesis can be broadly divided into three parts: (1) A study of projection algorithms, focussing on the practical case of non-convex constraints, (2) application of projection algorithms to the reconstruction of a binary image using its diffraction magnitudes, (3) use of projection algorithms to reconstruct a symmetric image from its diffraction magnitudes. Review material is presented in Chapter 1, and original work in Chapters 2-6.

Chapter 1 contains a review of background material relevant to this thesis. It covers the terminology and basic results used in the thesis, x-ray crystallography, iterative projection algorithms with a list of common iterative projection algorithms and crystallographic constraints, and commonly used error metrics.

Chapter 2 presents results concerning various aspects of the properties and practical applications of projection algorithms. A statistical model of the convergence of projection algorithms is described along with examples and an application of the model. A study of the behaviour of the algorithm when in the convergence region near the solution is then presented. This is followed by short discussions on the desirable properties of IPAs, the effects of uniqueness and false solutions, methods for handling more than two constraints, bootstrapping, estimating the solution and the effects of noise. Possible methods for improving the performance of the algorithm for very noisy data are described, followed by insights that can be obtained by writing projection algorithms as a map.

Chapter 3 discusses the reconstruction of periodic binary images from their diffraction magnitudes. Practical projections are developed for this problem and used in a reconstruction algorithm. The algorithm is applied to synthetic problems in 2D and the nature of the problem is explored.

Chapter 4 applies the methods developed in Chapter 3 to the reconstruction of molecular envelopes from crystal solvent contrast diffraction data. Appropriate modifications are made to the algorithm to take experimental realities into account.

Chapter 5 introduces the concepts of local and global symmetry and their effects on image reconstruction. Symmetry projections and the effects of interpolation on IPAs are assessed.

In Chapter 6, the methods described in Chapter 5 are applied to *ab initio* reconstruction of macromolecules with symmetry. Results are presented for an icosahedral virus using experimental x-ray diffraction data. Results are then presented for a protein molecule including determination of the location of the local symmetry operators.

Aspects of the work presented in this thesis have been published and presented. They are listed here in order of presentation.

V. L. Lo and R. P. Millane. Iterative projection algorithms with a binary constraint. In Proc. of Image and Vision Computing New Zealand, 2007. Best Student Poster Award.

V. L. Lo and R. P. Millane. Iterative projection algorithms for reconstructing compact binary images. In Image Reconstruction from Incomplete Data V, Proceedings of the SPIE, volume 7076, pages 70760B-70760B9, 2008.

V. L. Lo and R. P. Millane. Reconstruction of compact binary images from limited Fourier amplitude data. J. Opt. Soc. Am. A, 25:2600-2607, 2008.

V. L. Lo, R. L. Kingston, and R. P. Millane. Determination of molecular envelopes from solvent contrast variation data. Acta Cryst. A, 65:312-318, 2008.

R. P. Millane and V. L. Lo. Image reconstruction for crystalline and non-crystalline coherent diffractive imaging. ARC Centre of Excellence for Coherent X-ray Science, 3rd Annual workshop, 2008. Invited talk.

V. L. Lo, R. L. Kingston, and R. P. Millane. Reconstruction of macromolecular envelopes from crystal x-ray diffraction amplitudes. In IEEE International Conference on Image Processing, pages 2992-2995, Oct 2008.

V. L. Lo and R. P. Millane. Aspects of binary image reconstruction from Fourier amplitude data. In Proc. of Image and Vision Computing New Zealand, Nov 2008.

V. L. Lo and R. P. Millane. Ab initio determination of virus electron density from crystal diffraction data. ARC Centre of Excellence for Coherent X-ray Science, Satellite workshop, 2009.

V. L. Lo and R. P. Millane. Ab initio determination of virus electron density in x-ray crystallography. In OSA Signal Recovery and Synthesis, 2009.

V. L. Lo and R. P. Millane. Image reconstruction using symmetry. In Image Reconstruction from Incomplete Data VI," P.J. Bones, M.A. Fiddy and R.P. Millane (Eds.), Proc. SPIE, Vol. 7800, 78000N/1–9, 2010.

Chapter 1

Introduction

X-ray crystallography is a technique for determining the structure (positions of atoms in space) of molecules. It is a well developed technique, and little difficulty is encountered when imaging small molecules. However, problems still exist with the large molecules found in the increasingly important field of molecular biology. The recent rapid improvement in computational power has allowed more complex computational techniques to be used, reducing the need for expensive experiments. An area of particular promise has been the introduction of iterative projection algorithms. These are general algorithms which can be used in many inverse problems, and application of these algorithms has been shown to be useful in crystallography. This thesis extends the use of some of the more sophisticated iterative projection algorithms for crystallography, with the ultimate aim of reducing the need for experimental data collection to an absolute minimum.

This chapter introduces the terminology and basic concepts behind x-ray crystallography and iterative projection algorithms, followed by a short discussion on error metrics.

1.1 Terminology

This thesis is mainly concerned with the reconstruction of a crystal electron density, which is a three-dimensional image. Furthermore, in order for the computations to be carried out, the 3-D image needs to be sampled. Many of the concepts are illustrated in two dimensions for easier pictorial representation, but the concepts apply to any number of dimensions. Note that the effectiveness of constraints may be dependent on the number of dimensions.

1.1.1 Images

Consider a continuous m-dimensional image

$$\mathbf{x} = x(\mathbf{t}) = x(t_1, t_2, ...t_m), \tag{1.1}$$

where t_j are real numbers and **x** denotes the image. All practical images have *finite extent*, or *compact support*, i.e. are non-zero only within a finite region of space so that

$$x(t_1, t_2, ...t_m) = 0 \quad \text{for } t_j \notin (t_j^{min}, t_j^{max}) \tag{1.2}$$

where t_j^{min} and t_j^{max} define the support of the image.

An image is *periodic* with periods $\mathbf{L} = (L_1, L_2, ..., L_m)$ if it satisfies

$$x(t_1, t_2, ..., t_m) = x(t_1 + k_1 L_1, t_2 + k_2 L_2, ..., t_m + k_m L_m),\ \forall k_j \in \mathbb{Z}, \tag{1.3}$$

where the k_j are any integers. A periodic image is defined by one period of the image, and the support of one period is called the *unit cell*, which can be chosen to be anywhere in the image. Most commonly, the unit cell is defined in the interval $(-\mathbf{L}/2, \mathbf{L}/2)$ or $(0, \mathbf{L})$. Note that one period of a periodic image has compact support, but a periodic image does not, i.e. it is infinite in extent.

1.1.2 Sampling

In order for computations to be carried out, the continuous image $x(t_1, t_2, ..., t_m)$ needs to be sampled, with the samples generally arranged on a rectilinear grid. For an image of compact support, the sampled (or discrete) image, denoted $\mathbf{x} = x[t_1, t_2, ..., t_m]$ where the t_j are integers, is given by

$$x[t_1, t_2, ..., t_m] = x(s_1 t_1, s_2 t_2, ..., s_m t_m), \tag{1.4}$$

where the $\mathbf{s} = (s_1, s_2, ..., s_m)$ are the sample spacings and the discrete unit cell lengths are given by $N_j = L_j / s_j$. Then t_j is usually defined from $0 \leq t_j \leq N_j - 1$ or $-N_j/2 < t_j \leq N_j/2$.

The number of samples (pixels) is then the finite number $N = N_1 N_2 ... N_m$. The sampled image can then be represented as a vector $\mathbf{x} = (x_1, x_2, ..., x_N)$ of dimension N, with each element of **x** mapped to a sample/pixel in the image, and the value of the element set to the value of the pixel. A real image can therefore be represented as a point in an N-dimensional vector space $\mathbb{R}^N$. If the image is complex-valued, then a $2N$-dimensional vector space $\mathbb{R}^{2N}$ is required.

Sampling of a continuous image can be written as the multiplication of the image by the comb function $\kappa(t_1/s_1, t_2/s_2, ..., t_m/s_m)$, i.e.

$$x[t_1, t_2, ..., t_m] = x(t_1, t_2, ..., t_m)\kappa(t_1/s_1, t_2/s_2, ..., t_m/s_m). \tag{1.5}$$

where the comb function is defined by

$$\kappa(t_1, ..., t_m) = \sum_{t'_1=-\infty}^{\infty} \sum_{t'_2=-\infty}^{\infty} ... \sum_{t'_m=-\infty}^{\infty} \delta(t_1 - t'_1, t_2 - t'_2, ..., t_m - t'_m), \tag{1.6}$$

with $\delta(\cdot)$ denoting the Dirac delta function.

1.1.3 Fourier transform

The continuous Fourier transform of a continuous 1-dimensional image $\mathbf{x} = x(t)$ is defined by

$$\mathcal{F}\{\mathbf{x}\} = \mathbf{X} = X(h) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi th} dt, \tag{1.7}$$

where $\mathcal{F}\{\cdot\}$ denotes the Fourier transform operator, $i = \sqrt{-1}$, and h is the spatial frequency in Fourier space. The inverse continuous Fourier transform is defined by

$$\mathbf{x} = \mathcal{F}^{-1}\{\mathbf{X}\} = x(t) = \int_{-\infty}^{\infty} X(h) e^{i2\pi th} dh. \tag{1.8}$$

The Fourier transform of a continuous m-dimensional image $\mathbf{x} = x(t_1, t_2, ..., t_m)$ is defined by

$$\begin{aligned} \mathcal{F}\{\mathbf{x}\} &= \mathbf{X} = X(\mathbf{h}) = X(h_1, ..., h_m) \\ &= \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} x(t_1, ..., t_m) e^{-i2\pi t_1 h_1} ... e^{-i2\pi t_m h_m} \, dt_1 \, ... \, dt_m, \end{aligned} \tag{1.9}$$

where $\mathbf{h} = (h_1, h_2, ..., h_m)$ indexes the spatial frequencies. This is equivalent to a sequence of 1-D Fourier transforms along each dimension. The inverse Fourier transform is defined by

$$\begin{aligned} \mathcal{F}\{\mathbf{X}\} &= \mathbf{x} = x(\mathbf{t}) = x(t_1, ..., t_m) \\ &= \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} X(h_1, ..., h_m) e^{i2\pi t_1 h_1} ... e^{i2\pi t_m h_m} \, dh_1 \, ... \, dh_m. \end{aligned} \tag{1.10}$$

This is equivalent to a sequence of 1-D inverse Fourier transforms along each dimension.

Using Eq. (1.26), the sampling equation (1.5) can be written in the Fourier domain as

$$\begin{aligned} X'(\mathbf{h}) &= X(\mathbf{h}) \otimes \mathcal{F}\{\kappa(\mathbf{t}/\mathbf{s})\} \\ &= X(\mathbf{h}) \otimes \kappa(\mathbf{hs}), \end{aligned} \tag{1.11}$$

where $\otimes$ denotes convolution which is defined later in Eq. 1.22. This is because the Fourier transform of the comb function is itself, i.e. $\mathcal{F}\{\kappa(\mathbf{t})\} = \kappa(\mathbf{h})$. $X'(\mathbf{h})$ is the Fourier transform of the sampled image $x[\mathbf{t}]$, and can be seen to be a periodic image. The converse is also true, so that a periodic image will have a sampled Fourier transform.

If a continuous image is *bandlimited*, i.e.

$$X(\mathbf{h}) = 0 \quad \text{for } h_j > \frac{1}{2s'_j}, \quad j = 1, ..., m, \tag{1.12}$$

then if **x** is sampled with sampling spacing $s_j < s'_j$ in each dimension, the convolution in Eq. 1.11 does not cause any overlap, and the image can be perfectly reconstructed using sinc interpolation [1].

For a sampled image, either of finite extent or periodic, the discrete Fourier transform (DFT) is the discrete function,

$$\mathcal{F}\{\mathbf{x}\} = X[\mathbf{h}] = X[h_1, ..., h_m] = \sum_{t_m=0}^{N_m-1} ... \sum_{t_1=0}^{N_1-1} x[t_1, ..., t_m] e^{-i2\pi t_1 h_1} ... e^{-i2\pi t_m h_m}. \tag{1.13}$$

Consider sampling a continuous image $x(\mathbf{t})$ defined in $(-\mathbf{L}/2, \mathbf{L}/2)$ at sample spacing **s** to create the sampled image $x[\mathbf{t}]$ of size $\mathbf{N} = \mathbf{L}/\mathbf{s}$ pixels. The DFT of $x[\mathbf{t}]$ is then a discrete function $X[\mathbf{h}]$ with sample spacing $1/\mathbf{L}$ and with a highest frequency of $1/\mathbf{s}$, with a total size of **N** pixels also, i.e.

$$X[h_1, ..., h_m] = X(h_1/s_1, ..., h_m/s_m). \tag{1.14}$$

The inverse DFT is given by

$$\mathcal{F}\{\mathbf{X}\} = x[\mathbf{t}] = x[t_1, ..., t_m] = \sum_{h_m=0}^{N_m-1} ... \sum_{h_1=0}^{N_1-1} x[t_1, ..., t_m] e^{i2\pi t_1 h_1} ... e^{i2\pi t_m h_m}. \tag{1.15}$$

If an image is real (zero imaginary part), then its Fourier transform is *Hermitian*, so that

$$\begin{aligned} X[h_1, h_2, ..., h_m] &= X^*[-h_1, -h_2, ..., -h_m] & (1.16) \\ &= Re\{[-h_1, -h_2, ..., -h_m]\} - i\, Im\{X[-h_1, -h_2, ..., -h_m]\} & (1.17) \end{aligned}$$

where * denotes the complex conjugate, and $Re\{\cdot\}$ and $Im\{\cdot\}$ are the real and imaginary part operators respectively. Two such related points are referred to as a *Friedel pair*.

The RMS difference between any two images is the same in image space as it is in Fourier space, that is,

$$\sum_{\mathbf{t}} (x[\mathbf{t}] - y[\mathbf{t}])^2 = \sum_{[\mathbf{h}]} (\mathcal{F}\{x\}[\mathbf{h}] - \mathcal{F}\{y\}[\mathbf{h}])^2 \tag{1.18}$$

where **x** and **y** are two images, and $\mathcal{F}\{x\}[\mathbf{h}]$ denotes the value at pixel $[\mathbf{h}]$ of the Fourier transform of **x**, and the summation is over every element in the image or its DFT. In the Euclidean space $\mathbb{R}^{2N}$, the distance between two images (points) is the same, and so the Fourier transform can be thought of as a rotation of the axes, or a new basis function.

The Fourier transform of a 3D sampled image circularly shifted by $\mathbf{d} = (d_1, d_2, d_3)$ is

$$\mathcal{F}\{x[t_1 + d_1, t_2 + d_2, t_3 + d_3]\} = X[h_1, h_2, h_3]\, e^{i2\pi d_1/N_1}\, e^{i2\pi d_2/N_2}\, e^{i2\pi d_3/N_3}, \tag{1.19}$$

where the d_i are any integers. The circular shift results in a phase shift but does not change the Fourier magnitudes.

The Fourier transform for an image inverted in the origin is given by

$$\mathcal{F}\{x[-t_1, -t_2, -t_3]\} = X[-h_1, -h_2, -h_3]. \tag{1.20}$$

Using Eqs. (1.16) and (1.20) shows that for a real image $\mathcal{F}\{x(-t_1, -t_2, -t_3)\} = X^*[h_1, h_2, h_3]$ so that the phases of the Fourier transform are changed to their negatives but the Fourier magnitudes are left unchanged.

The Fourier transform of the negative of an image is given by

$$\mathcal{F}\{-x[t_1, t_2, t_3]\} = -X[h_1, h_2, h_3], \tag{1.21}$$

so that changing the sign of an image leaves the Fourier magnitudes unchanged.

1.1.4 Multiplication, convolution, and correlation

If two images **x** and **y** are the same size, they can be multiplied element-wise to form a third image **xy**.

The convolution of two continuous images **x** and **y** is defined by

$$\mathbf{x} \otimes \mathbf{y} = \mathbf{y} \otimes \mathbf{x} \tag{1.22}$$

$$= \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} x(t'_1, ..., t'_m) y(t_1 - d_1, ..., t_m - d_m), dt'_1 ... dt'_m. \tag{1.23}$$

Convolution is equivalent to multiplication in the Fourier domain, i.e.

$$\mathcal{F}\{\mathbf{x} \otimes \mathbf{y}\} = \mathcal{F}\{\mathbf{x}\}\mathcal{F}\{\mathbf{y}\}. \tag{1.24}$$

Correlation is equivalent to convolution with one of the images inverted in the origin, i.e.

$$\mathbf{x} \odot \mathbf{y} = x(t_1, ..., t_m) \otimes y(-t_1, ..., -t_m), \tag{1.25}$$

and

$$\mathcal{F}\{\mathbf{x} \odot \mathbf{y}\}(\mathbf{h}) = \mathcal{F}\{\mathbf{x}\}(\mathbf{h})\mathcal{F}\{\mathbf{y}\}(-\mathbf{h}). \tag{1.26}$$

The correlation of an object with itself is called the *autocorrelation*. The maximum value of

the autocorrelation is always at the origin. In the Fourier domain, the autocorrelation of a real image **x** is given by

$$\begin{aligned}\mathcal{F}\{\mathbf{x}\odot\mathbf{x}\}(\mathbf{h}) &= \mathcal{F}\{\mathbf{x}\}(\mathbf{h})\mathcal{F}\{\mathbf{x}\}(-\mathbf{h}) \\ &= \mathcal{F}\{\mathbf{x}\}(\mathbf{h})\mathcal{F}\{\mathbf{x}\}^*(\mathbf{h}) \quad \text{if } \mathbf{x} \text{ is real} \\ &= |\mathcal{F}\{\mathbf{x}\}(\mathbf{h})|^2. \end{aligned} \tag{1.27}$$

The convolution of two discrete images **x** and **y** is defined by

$$\mathbf{x}\otimes\mathbf{y} = \mathbf{y}\otimes\mathbf{x} \tag{1.28}$$

$$= \sum_{t'_1=-\infty}^{\infty}\sum_{t'_2=-\infty}^{\infty}\ldots\sum_{t'_m=-\infty}^{\infty} x[t'_1,...,t'_m]y[t_1-t'_1,...,t_m-t'_m] \tag{1.29}$$

where $\otimes$ denotes convolution. If the two images are periodic with the same period, the result of the convolution or correlation will also be periodic with the same period. This is known as *circular convolution*. Circular convolution is equivalent to multiplication of the DFTs of the images as in Eq. (1.26), and correlation is equivalent to convolution with one of the images inverted in the origin i.e.

$$\mathbf{x}\odot\mathbf{y} = x[t_1,...,t_m]\otimes y[-t_1,...,-t_m]. \tag{1.30}$$

Similar to the result for a continuous image, the Fourier transform of the autocorrelation of a discrete real image is given by

$$\mathcal{F}\{\mathbf{x}\odot\mathbf{x}\}[\mathbf{h}] = |\mathcal{F}\{\mathbf{x}\}[\mathbf{h}]|^2. \tag{1.31}$$

1.2 X-ray crystallography

X-ray crystallography is a technique for determining the structure (positions of atoms in space) of molecules [2]. A beam of collimated, monochromatic x-rays is directed at a crystal of the molecule under study and the resulting diffraction pattern is measured. The diffraction data are inverted numerically to calculate the electron density from which the positions of the atoms, i.e. the structure of the molecule, can be inferred. X-ray crystallography is a well-developed technique, with the first diffraction of x-rays from crystals being found by Friedrich and Kipling in 1912, and is currently the most powerful technique for structure determination of both organic and inorganic molecules.

X-rays are electromagnetic radiation with wavelengths between 0.1 and 100Å. Usually single-wavelength collimated beam of x-rays is used for x-ray crystallography experiments. It can be produced by directing high energy electrons at a metallic target to produce a multiwavelength set of electromagnetic radiation, which is then filtered or diffracted to select

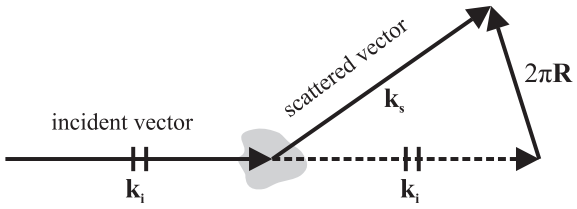


Figure 1.1 **R** is the vector difference between the incident and scattered wavevectors.

a single wavelength. X-rays crystallography experiments at lower energies can often be done in a laboratory, but higher intensities require the use of a synchrotron [2]. The resultant electromagnetic waves are *collimated* i.e. made unidirectional by using a collimator.

The x-rays are directed at a crystalline sample, and the x-ray photons scatter off the electrons in the atoms. To a very good approximation an x-ray photon is scattered by at most one electron. The scattering is then "weak", and this is known as the Born approximation. It is also assumed that the diffraction pattern is measured far from the sample, so that the size of the sample is small compared to the distance to the detector. This is known as the far field, or Fraunhofer, approximation. Then at the detector, the complex diffracted amplitude is given by

$$A(\mathbf{R}) = \int_{-\infty}^{\infty} \rho_e(\mathbf{r}) e^{i2\pi\mathbf{R}.\mathbf{r}} d\mathbf{r} \tag{1.32}$$

where $\rho_e(\mathbf{r})$ is the electron density of the sample and $\mathbf{R}$ is the scattering vector which indexes Fourier space, usually referred to as *reciprocal* or *diffraction* space in crystallography. An alternative interpretation is to consider an incident wavevector given by $\mathbf{k}_i = 2\pi/\lambda \mathbf{s}_i$, where λ is the wavelength and $\mathbf{s}_i$ is a unit vector in the direction of propagation. The incident wave scatters off the sample to give the scattered wave with wavevector $\mathbf{k}_s$. Then $\mathbf{R}$ is the vector difference between the incident and scattered wavevector as show in Fig. 1.1. Eq. (1.32) is a Fourier transform, identical to Eq. (1.9), so that in the far field, the diffracted amplitudes are the Fourier transform of the electron density. For a particular crystal orientation in the beam $A(\mathbf{R})$ is measured on a sphere, and if multiple orientations of the crystal are used, the 3D function $A(\mathbf{R})$ can be built up. Only the magnitudes of the diffracted x-rays, or the Fourier transform, can be measured. Therefore, Eq. (1.32) cannot be directly inverted to calculate the electron density. The phases must be found using some additional information. This is known as a *phase retrieval* problem, and is found in diverse fields such as crystallography, astronomy, and medical imaging [3]. Despite the name, the primary aim in most phase retrieval problems is to find the image domain representation of the object. The Fourier phases themselves are almost never the primary objective.

If the object of interest is not periodic (e.g. a single molecule), then the Fourier transform is continuous, and so the Fourier magnitudes can be sampled as finely as desired. Consider an image $\mathbf{x}$ which has known support $(-\mathbf{L}/2, \mathbf{L}/2)$. If the Fourier magnitudes are measured

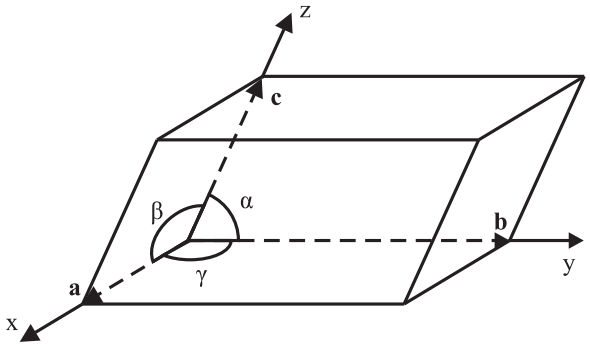


Figure 1.2 A crystallographic unit cell in the crystal lattice.

at sampling distance $1/\mathbf{L}' < 1/\mathbf{L}$, then the inverse DFT of the Fourier magnitudes will result in an image which is known to be zero from **L** to $\mathbf{L}'$. These zeros can then be exploited with a support constraint to help phase the object. If $\mathbf{L}' > 2\mathbf{L}$, and all the Fourier magnitudes are known, then the number of data is equal to the number of pixels in the image, and in principle the image can be reconstructed using only the Fourier magnitudes. The major difficulty with this approach is in collecting enough signal to accurately measure the diffraction magnitudes, since the scattering from a single small object is very weak. Use of a very intense x-ray beam improves the signal level but also damages the sample. Nevertheless, use of intense x-ray sources and single particles is an active research area [4, 5, 6].

1.2.1 Crystal periodicity

The solution to the signal-to-noise problem described above is to use a *crystal*. In a crystal, the molecules are arranged in a regular periodic lattice. Each periodic unit is called a *unit cell*. The unit cell is described using three vectors **a**, **b**, **c** on the edges of each unit cell with angles α, β and γ between them as shown in Fig. 1.2. Each point **r** can be then be referenced by

$$\mathbf{r} = x\mathbf{a} + y\mathbf{b} + z\mathbf{c}, \tag{1.33}$$

with $0 \leq x, y, z \leq 1$ if **r** is inside the unit cell with one corner at the origin.

The periodic nature of the image can be represented as a convolution of the electron density in one unit cell with the periodic lattice consisting of an array of Dirac delta functions, i.e.

$$\rho_c(\mathbf{r}) = \rho_{cell}(\mathbf{r}) \otimes \ell(\mathbf{r}) \tag{1.34}$$

where the lattice function $\ell(\mathbf{r})$ is defined as

$$\ell(\mathbf{r}) = \kappa(\mathbf{r}/\mathbf{L}) = \sum_m \sum_n \sum_p \delta(\mathbf{r} - (m\mathbf{a} + n\mathbf{b} + p\mathbf{c})) \tag{1.35}$$

for all integers m, n, p. Then using the convolution theorem Eq. (1.26) the complex diffracted amplitudes are given by

$$A(\mathbf{R}) = \varrho_{cell}(\mathbf{R})\mathcal{L}(\mathbf{R}), \tag{1.36}$$

where $\varrho_{cell}(\mathbf{R})$ and $\mathcal{L}(\mathbf{R})$ are the Fourier transforms of $\rho_{cell}(\mathbf{r})$ and $\ell(\mathbf{r})$ respectively. $\mathcal{L}(\mathbf{R})$ is also a lattice and is known as the *reciprocal lattice*, and can be written as

$$\mathcal{L}(\mathbf{R}) = \kappa(\mathbf{r}/\mathbf{L}) = \sum_h \sum_k \sum_l \delta(\mathbf{R} - (h\mathbf{a}' + k\mathbf{b}' + l\mathbf{c}')), \tag{1.37}$$

where h, k and l are integers that index the reciprocal lattice. The reciprocal lattice vectors a', b', and c' are defined by

$$a' = \frac{\mathbf{b} \times \mathbf{c}}{|\mathbf{a} \cdot \mathbf{b} \times \mathbf{c}|} \tag{1.38}$$

$$b' = \frac{\mathbf{c} \times \mathbf{a}}{|\mathbf{a} \cdot \mathbf{b} \times \mathbf{c}|} \tag{1.39}$$

$$c' = \frac{\mathbf{a} \times \mathbf{b}}{|\mathbf{a} \cdot \mathbf{b} \times \mathbf{c}|}, \tag{1.40}$$

where $\times$ denotes the vector cross product. Note that in crystallography a', b' and c' are usually denoted a^*, b^* and c^* but the latter notation is used here for the complex conjugate.

Referring to Eq. (1.36), the diffraction amplitudes $A(\mathbf{R})$ of a crystal are zero except at the points of the reciprocal lattice. These correspond to diffraction peaks on the detector which are known as *Bragg peaks*. The signals at the Bragg peaks are the direct summation of the signals from each of the unit cells, and so the signal-to-noise ratio is much improved.

The Bragg peaks sample the Fourier transform at the Nyquist rate [3], which is the rate at which if the phases at the peaks were known, then exactly enough information is available to reconstruct the electron density using a simple inverse Fourier transform. However, the phases are not known, so there is insufficient information available to uniquely reconstruct the solution. Furthermore, fine sampling of the Fourier magnitudes is no longer possible since the signal is only available at the Bragg peaks. Therefore, ancillary data must be used. Some ancillary data such as the total mass and volume of the molecule can easily be obtained, but most of the more useful ancillary data require additional experiments to be performed.

If the crystal is small then the Bragg peaks are broader. Define the shape function of the crystal $s(\mathbf{r})$ to be 1 inside the boundary of the crystal and 0 outside. Then the electron density of the finite crystal $\rho_{fin}(\mathbf{r})$ is

$$\rho_{fin}(\mathbf{r}) = \rho_{cell}(\mathbf{r}) \otimes (\ell(\mathbf{r})s(\mathbf{r})), \tag{1.41}$$

and in the Fourier domain,

$$A_{fin}(\mathbf{R}) = \varrho_{cell}(\mathbf{R})(\mathcal{L}(\mathbf{R}) \otimes S(\mathbf{R})) \quad (1.42)$$
$$= A(\mathbf{R}) \otimes S(\mathbf{R}). \quad (1.43)$$

where $S(\mathbf{R})$ is the Fourier transform of the shape function. Therefore, each Bragg peak is convolved with $S(\mathbf{R})$, and as the crystal (or more specifically, the size of the optically coherent domains) get larger, the Bragg peaks get sharper since the size (support) of $S(\mathbf{R})$ is inversely proportional to that of $s(\mathbf{R})$. If $S(\mathbf{R})$ is not too large as to significantly extend into the adjacent Bragg peaks, then it is still possible to measure the individual peak amplitudes.

The complex Fourier amplitudes are usually denoted F_{hkl} where h, k and l are integers, i.e. $F_{hkl} = A(ha', kb', lc')$ and are referred to as the *structure factors*. For computational purposes, a 3-D grid is generally created with the value of F_{hkl} at each grid point. The grid is large enough such that all the structure factors can be inserted. If the phases at each grid point were known, then the inverse discrete Fourier transform could be applied, giving a 3-D grid of the same size which samples the electron density. The sampling distance, or resolution, of the grid is directly dependent upon the maximum spatial frequency of the Fourier magnitudes, which is in turn equivalent to the size of the grid.

The autocorrelation of the electron density can be found by taking the inverse Fourier transform of the square of the Fourier magnitudes as shown in Eq. (1.27). This is referred to as the *Patterson function* in crystallography and is commonly used for finding basic information about the crystal such as its crystallographic symmetry group. For small molecules, it is possible to find the positions of the atoms directly from the Patterson function .

Standard crystallographic notation has been used in this section. In the rest of the thesis, in order to conserve variables and keep the techniques described easily generalizable to different numbers of dimensions, images are denoted $x(\mathbf{t}) = x(t_1, t_2, ..., t_m)$, with a corresponding Fourier transform of $X(\mathbf{h}) = X(h_1, h_2, ..., h_m)$ for a unit cell of size $\mathbf{L} = (L_1, L_2, ..., L_m)$. Sampled images are denoted $x[\mathbf{t}] = x[t_1, t_2, ..., t_m]$ and $X[\mathbf{h}] = X[h_1, h_2, ..., h_m]$ in the image and Fourier domains respectively. It is assumed that the unit cell is orthorhombic, i.e. $\alpha = \beta = \gamma = 90°$. If necessary, the techniques described can easily be extended to non-orthorhombic unit cells.

1.2.2 Characteristics of crystallographic data

The majority of the x-ray photons pass through the crystal without scattering, so there is a strong undeflected beam at the detector. Therefore, a *beamstop* must be placed in the centre of the detector to prevent saturation and damage, and this prevents collection of the origin term of the Fourier transform and possibly some of the other low resolution terms. Various methods are used to minimize the size of the beamstop, but in general a few of the low order reflections cannot be measured. This can pose a problem since, like most

natural images, much of the energy, and therefore information, in the Fourier transform is contained in the lower resolution terms. Furthermore, information on the general shape of the molecule is contained in these low order magnitudes. Also, diffraction data are collected only up to some maximum isotropic resolution due to the limited size of the detector and also due to limited signal-to-noise ratio (SNR) at high resolution.

In most cases the electron density can be considered to be a purely real function, making its Fourier transform Hermitian, and the Fourier magnitudes centrosymmetric. Note that the inverse applies, so a centrosymmetric object will have a purely real Fourier transform, making its reconstruction somewhat easier.

1.2.3 Symmetry

In many crystals, there may be multiple copies of the molecule (or identical parts of a molecule, or subunits) in the unit cell. This introduces an additional symmetry to the translational unit cell lattice symmetry. A symmetry with q repeated subunits has *order* q, or can also be said to have a q-fold symmetry. Symmetry can be divided into two types: *crystallographic* and *non-crystallographic* symmetry. With crystallographic symmetry, the entire periodic crystal satisfies the symmetry, and with non-crystallographic symmetry only a restricted region satisfies the symmetry.

The coverings of 3D space with unit cells in a periodic lattice can be classified into 7 crystal classes. Within these there are a number of possible crystallographic symmetries that lead to a total of 230 possible crystallographic symmetries, called space groups, which are listed in the International Tables for Crystallography [7]. A common convention of referring to the symmetries is with a letter and number combination. The first letter represents the lattice, and the numbers are the orders of the rotation, screw and reflection axes. For example, a common symmetry is the order 4 $P2_12_12_1$ symmetry. The P indicates that it is a primitive unit cell where there is only one lattice point per unit cell. The 2_1 represents a 2-fold screw axes, for which the basic symmetry operation is a π rotation followed by translation parallel to the axis of rotation of half of the translational repeat of the lattice. A crystal with no crystallographic symmetry has symmetry group $P1$. Although the work done in this thesis is with rectilinear crystals where all three axes are orthogonal to each other, the techniques developed can easily be extended to non-rectilinear systems.

The presence of order q symmetry means that the number of parameters needed to describe the object is reduced by a factor q. However, when the crystallographic symmetry is expressed in the Fourier domain it causes the number of independent Fourier magnitudes to be reduced by the same order (assuming there are no anomalous scatterers). For example, using the $P2_12_12_1$ symmetry as an example, the symmetry operators in image space

are given by

$$x(t_1, t_2, t_3) = x(-t_1 + 0.5L_1, -t_2, t_3 + 0.5L_3) \quad (1.44)$$
$$x(t_1, t_2, t_3) = x(-t_1, t_2 + 0.5L_2, -t_3 + 0.5L_3) \quad (1.45)$$
$$x(t_1, t_2, t_3) = x(t_1 + 0.5L_1, -t_2 + 0.5L_2, -t_3). \quad (1.46)$$

Then in Fourier space,

$$X(h_1, h_2, h_3) = X(-h_1, -h_2, h_3)e^{i\pi h_1}e^{i\pi h_3} \quad (1.47)$$
$$X(h_1, h_2, h_3) = X(-h_1, h_2, -h_3)e^{i\pi h_2}e^{i\pi h_3} \quad (1.48)$$
$$X(h_1, h_2, h_3) = X(-h_1, -h_2, -h_3)e^{i\pi h_1}e^{i\pi h_2}. \quad (1.49)$$

For example, referring to Eq. (1.47) shows that $X(h_1, h_2, h_3) = \pm X(-h_1, -h_2, h_3)$. Since the magnitude of these two reflections is equal, the number of data has been halved.

Since each operator in image space is related to another operator in Fourier space, the number of parameters and data are reduced by the same degree and no extra information is gained from the crystallographic symmetry. The symmetry can be used however, to gain some efficiencies in the representation and reconstruction of the crystal structure, as well as increasing the signal to noise ratio by averaging. It is possible to determine the crystallographic symmetry of the molecule by inspection of the Fourier magnitudes.

Non-crystallographic symmetry (NCS) applies only over a restricted domain of the unit cell. NCS does not reduce the number of independent Fourier data and thus provides extra information. NCS has been used extensively in x-ray crystallography and is discussed in detail in Chapters 5 and 6.

1.2.4 Inversion methods in crystallography

The inverse problem in x-ray crystallography is to determine the electron density from the Fourier magnitudes and other ancillary information. The problem corresponds to a phase problem. A number of methods are used, depending on the circumstances. The key techniques are outlined in the following subsections.

1.2.4.1 Direct methods

The experimentally least complex method is to use what are referred to as *direct methods*, which rely upon atomicity, i.e. the electron density consisting of sharp, separated peaks. If sufficiently high resolution data of around 1.4Å resolution or better can be collected, the electron density peaks associated with each atom are resolved. In this case the atomic coordinates can be treated as parameters and the number of diffraction amplitudes exceeds the number of parameters by a wide margin. This large data/parameter ratio is exploited probabilistically in direct methods algorithms that allow the phases to be determined directly [8]. Direct methods are effective only for molecules with less than around 1000 non-

hydrogen atoms. They are not effective for large molecules because of the complexity of the large number of atoms and also because it is much more difficult to collect high (atomic) resolution data.

1.2.4.2 Isomorphous replacement

In the isomorphous replacement method, heavy atoms are added to the crystal which bind to identical sites on all the molecules. It is assumed that the structure is not significantly changed by the addition of the atoms, i.e. if there is *non-isomorphism* the technique will not work. The cross vectors between heavy atoms show up as large peaks in the Patterson function, from which their positions in the unit cell can be inferred. Using diffraction data from the native molecule and a number of heavy atom derivatives with known heavy atom positions, the phases of the structure factors can be determined [2].

1.2.4.3 Anomalous scattering

Atoms exhibit *anomalous scattering* when the wavelength of the x-rays is near an absorbtion edge, which are at specific energies for each element. The effect of anomalous scattering is to add an additional complex component to the scattering. Thus the total scattering for a single anomalously scattering atom can be written

$$f_{anom} = f + \delta f + jf'' \tag{1.50}$$
$$= f' + jf'' \tag{1.51}$$

where f is the normal scattering, and $\delta f + jf''$ is the additional anomalous scattering component. With the addition of anomalous scattering, the Fourier magnitudes are no longer Hermitian, i.e. $X(h_1, h_2, h_3) \neq X^*(-h_1, -h_2, -h_3)$.

The oxygen, hydrogen, and nitrogen atoms commonly found in biological molecules do not exhibit significant anomalous scattering at the x-ray wavelengths commonly used for crystallography. Therefore suitable anomalous scatterers are added in a way analogous to that for heavy atom isomorphous replacement. Selenomethionine (SeMet) derivatives are commonly used as the anomalous scatterers [2]. Diffraction data from native and anomalous scattering derivatives can be used to determine the phases in a similar manner as for isomorphous replacement.

An important extension to the anomalous scattering method is the method of multiwavelength anomalous dispersion (MAD) [9]. By changing the wavelength and collecting sets of data above and below an absorption edge, additional data are obtained that can be used to solve for the electron density. This is currently the most convenient and effective phasing technique.

1.2.4.4 Molecular replacement

If the structure of a similar molecule is known, it can be used as a starting point for determining the structure of the unknown molecule, significantly simplifying the phase problem since the phases need only to be refined rather than calculated from scratch. Since many protein structures have been determined, the structure of a similar molecule is often available. In general, the orientation and position of the unknown molecule in the unit cell is not the same as that for the known molecule, so the known structure must first be oriented and positioned in the unit cell of the unknown molecule. This can be time consuming but fast rotation and translation functions have been developed which reduce the time taken. With a reasonable model of the unknown structure now in place, the phases from the known structure are used as starting phases for the unknown structure. The phases are then refined using algorithms that incorporate the Fourier amplitude data of the unknown structure and other structural information [10].

1.2.4.5 Ab initio phasing

The holy grail of macromolecular crystallographic methods and a major focus of this thesis is the development of an *ab initio* method of phasing, where no additional data other than the diffraction data needs to be collected. As shown in Sec. 1.2.1, this is impossible for a general electron density function.

However, there is always additional information about the electron density that can be found with minimal effort. Firstly, general information about the crystal such as the molecular weight, the fraction of the unit cell occupied by the protein and the electron density of the solvent can be gathered without much effort. Secondly, assumptions may be made about the nature of the electron density such as the fact that electron density is always positive, proteins generally form a globular shape, and each unit cell is connected to its neighbours by some atoms to ensure the structural integrity of the crystal. Thirdly, since many crystals have non-crystallographic symmetry, the Patterson function may be searched for such a symmetry, and if found, the orientation of the non-crystallographic symmetry axes can be calculated directly from the Patterson function. Finally, it may be possible to exploit the fact that a protein is made up of folded chains of amino acids as a constraint, although this is extremely difficult and is not discussed in this thesis.

1.3 Iterative projection algorithms

An *inverse problem* is a common problem in many branches of science and mathematics. In an inverse problem, a set of parameters must be found based on some measurements (data) related to those parameters. The forward mapping from the parameter values to the data is easy to compute, but to find the values of the parameters from the measurements, i.e. the inverse operation, is difficult. The parameter values which need to be found is referred to here as an *image*. The most interesting, and difficult, inverse problems are those where

the data are "incomplete" i.e. are insufficient to uniquely define the image. In this case, additional information must be incorporated to solve the problem. Inverse problems can often be described as a set satisfaction problem, where one attempts to find the intersection between sets, with each set being the set of images which satisfy a particular *constraint*, and each constraint corresponds to a piece of information or data on the image.

An *iterative projection algorithm* (IPA) is an algorithm which attempts to reconstruct the image from the constraints using a series of *projections* onto the constraints. A projection is an operation on an image which makes the minimum RMS change to the image such that the new image satisfies the constraint. Each constraint can be represented in $\mathbb{R}^{2N}$ by the manifold of the set of points (images) which satisfy that constraint, and a projection onto the constraint set is identical to a projection in the geometrical sense of the term. Projections are more rigorously defined in Sec. 1.3.2.

IPAs are of most value when the constraints are disparate and not easily combined. A common situation is to have constraints in both the image and Fourier domains. The constraints in each of the domains are easily combined, but combining constraints from the two domains is not easy. Since the distance preserving property of the Fourier transform as described in Eq. (1.18) means that a projection in Fourier space is identical to a projection in image space, an IPA is an ideal way to combine the two sets of constraints. The solution for problems with more than two constraints is discussed further in Sec. 2.6 but in general, most IPAs accept only two constraints.

1.3.1 Constraint set geometry

The geometry of the constraint set in $\mathbb{R}^{2N}$ is important in terms of the ease with which the algorithm converges towards the solution. A *manifold* is usually defined as a set that on a small enough scale resembles a Euclidean space, and is used here to refer to the Euclidean space representation of a set. It can be seen from Eq. 1.18 that the shape of the constraint set manifold is identical in image and Fourier space. A key property of a constraint set is its *convexity*.

A *convex* set is defined as a set of points where all points on the line segment between any two points in the set is also inside the set. That is, if a and b are any two points in a convex constraint set C, then

$$(a + \eta(b - a)) \in C \quad \text{if } 0 \leq \eta \leq 1. \tag{1.52}$$

If a set is not convex, then it is known as a non-convex set. If D is a *convex constraint of infinite extent*, then η can be any scalar, i.e.

$$(a + \eta(b - a)) \in D \quad \forall\, \eta \in \mathbb{R}. \tag{1.53}$$

Any linear combination of points in such a constraint space will remain in the constraint space.

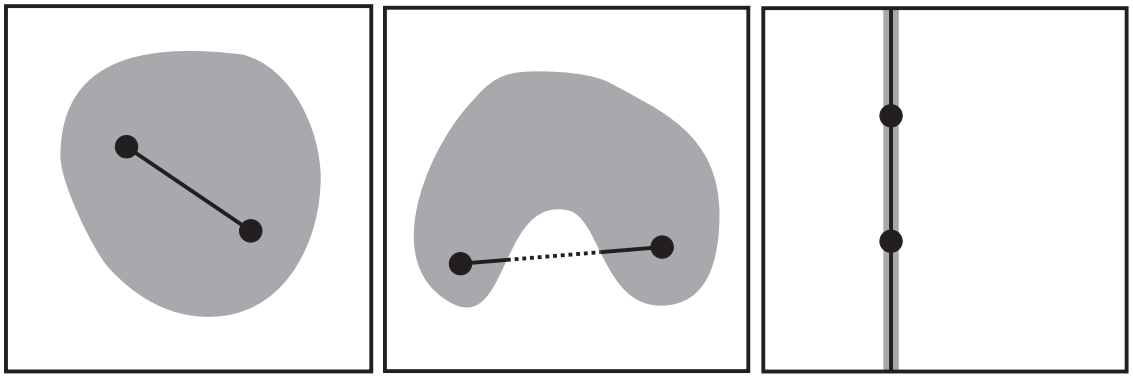


Figure 1.3 Constraint set shapes (a) Convex, (b) Non-convex, (c) Infinite convex.

Fig. 1.3 illustrates a convex constraint, a non-convex constraint, and a convex constraint of infinite extent in 2-D.

1.3.2 Projections

The projection of an image (a point in $\mathbb{R}^{2N}$) $\mathbf{x}$ onto a constraint A is denoted $P_A\mathbf{x}$ and is defined by

$$P_A\mathbf{x} = \underset{\mathbf{x}' \in A}{\operatorname{argmin}} \|\mathbf{x}' - \mathbf{x}\|, \tag{1.54}$$

where $\|\cdot\|$ is the Euclidean norm, and $\underset{\mathbf{x}}{\operatorname{argmin}}\,[\alpha(\mathbf{x})]$ denotes the value of $\mathbf{x}$ that minimizes $\alpha(\mathbf{x})$.

A projection is clearly an idempotent operation, i.e.

$$P_A P_A \mathbf{x} = P_A \mathbf{x}. \tag{1.55}$$

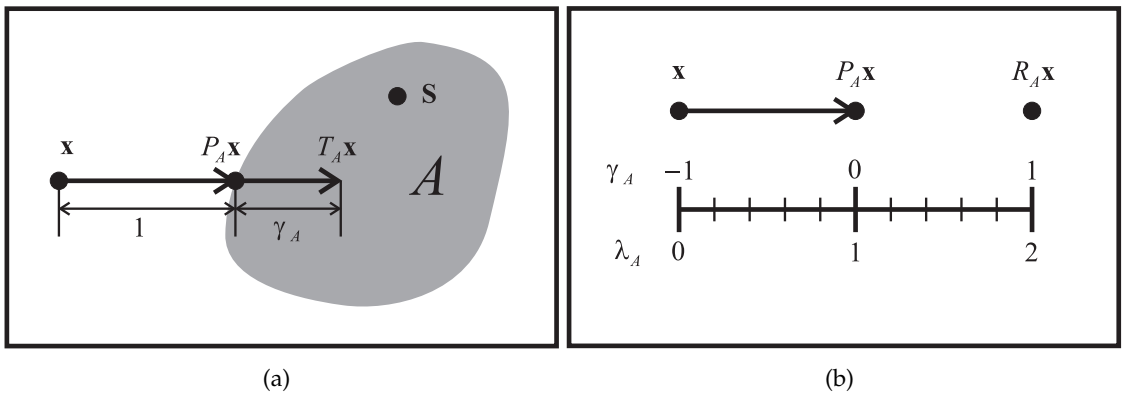
Relaxed projections are a useful way to generate a new point from a current point and its projection by using a linear combination of the two. They are a common theme in many projection algorithms. Define a *relaxed* projection $T_A\mathbf{x}$ by

$$T_A\mathbf{x} = P_A\mathbf{x} + \gamma_A(P_A\mathbf{x} - \mathbf{x}), \tag{1.56}$$

where γ_A is called the relaxation parameter. A diagram of a relaxed projection is shown in Fig. 1.4(a). Note that an alternative terminology for a relaxed projection is

$$T_A\mathbf{x} = \lambda_A P_A\mathbf{x} + (1 - \lambda_A)\mathbf{x}, \tag{1.57}$$

where $\lambda_A = \gamma_A + 1$. The relationship between γ_A and λ_A is shown diagrammatically in



(a) (b)

Figure 1.4 Relaxed projections (a) A relaxed projection onto a constraint set, (b) the relationship between the relaxation parameters γ_A and λ_A.

Fig. 1.4(b). A special case of a relaxed projection is a *reflected* projection $R_A\mathbf{x}$, which is when $\gamma_A = 1$ or $\lambda_A = 2$, i.e.

$$R_A\mathbf{x} = 2P_A\mathbf{x} - \mathbf{x}. \tag{1.58}$$

A projection or relaxed projection with $-1 < \gamma < 1$ onto a convex constraint set decreases the error to any point in the set (the solution). Consider a convex constraint set C and a point $\mathbf{x}$ in $\mathbb{R}^{2N}$ as shown in Fig. 1.5, and a solution $\mathbf{S} \in C$. The projection of the point $\mathbf{x}$ onto the set C is $P_C\mathbf{x}$, and the line $\mathbf{x} - P_C\mathbf{x}$ is normal to the surface of C at $P_C\mathbf{x}$. The $(N-1)$-dimensional hyperplane which is tangent to the boundary of C at $P_C\mathbf{x}$ is denoted V and partitions $\mathbb{R}^{2N}$ into two with all of C on the side of V not containing $\mathbf{x}$. Define the region of $\mathbb{R}^{2N}$ on the side of V containing C to be V_1. Then $C \subset V_1$. Denote the $(N-1)$-dimensional hyperplane which is normal to $P_C\mathbf{x} - \mathbf{x}$ at $0.5(T_C\mathbf{x} + \mathbf{x})$ as V'. V' is the locus of points equidistant from $T_C\mathbf{x}$ and $\mathbf{x}$. Then V and V' are parallel, and V_1 is wholly contained on the side of V' not containing $\mathbf{x}$. So $|T_C\mathbf{x} - \mathbf{S}| \leq |\mathbf{x} - \mathbf{S}|$ for all $\mathbf{S}$, with equality if and only if $\mathbf{x} = P_C\mathbf{x} \in C$. Therefore, the projection or relaxed projection (with $-1 < \gamma < 1$) reduces the distance (error) to the solution.

Thus if there is a set of convex constraints C_1, C_2, ..., C_k, then a series of projections or relaxed projections with $-1 < \gamma_A < 1$ onto the constraints in any order will lead to a monotonically decreasing distance from the solution. If a solution does exist, then convergence is guaranteed as long as all constraints are used a sufficient number of times.

1.3.3 Iterative projection algorithms

An iterative algorithm consists of repeatedly applying a *recursion* to an *iterate*, i.e.

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n). \tag{1.59}$$

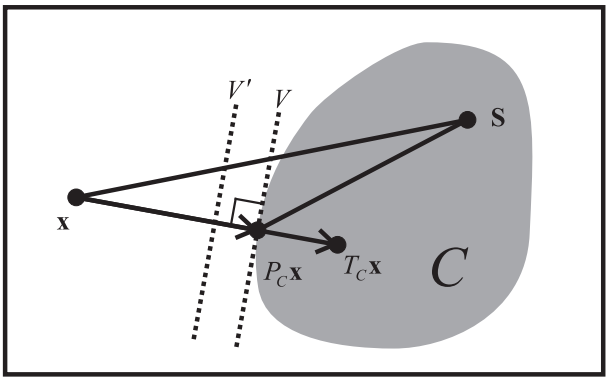


Figure 1.5 A projection or relaxed projection with $-1 < \gamma < 1$ onto a convex constraint set reduces the error. V denotes the hyperplane tangent to the boundary of C at $P_C\mathbf{x}$, and V' denotes the hyperplane which is the locus of points equidistant from $\mathbf{x}$ and $T_C\mathbf{x}$.

where $\mathbf{x}_n$ is the n^{th} iterate and $f(\cdot)$ is the update rule. Note that the iterate is sometimes, but not always, the estimate of the solution. An iterative algorithm *halts* if $\mathbf{x}_{n+1} = \mathbf{x}_n$, since no more progress will be made. Most algorithms halt when the solution is found, and an iterative algorithm is said to *stagnate* if the algorithm halts before a solution has been found.

If the algorithm is an iterative projection algorithm, the update rule consists of a combination of projections and the iterate. Generally, it is a composition of a linear combination of $P\mathbf{x}_n$ and $\mathbf{x}_n$. Some common IPAs will now be described. Many of these algorithms were designed for phase retrieval, where the Fourier magnitudes of the image are known and must be combined with some additional image domain constraints, most commonly the support and positivity constraints which are described in Sec. 1.3.4. They can be applied, however, to any inverse problem posed as a constraint satisfaction problem.

1.3.3.1 Error reduction algorithm

The simplest projection algorithm consists of alternating projections onto each of the constraint sets. It is sometimes known as the error reduction (ER) algorithm [11], or alternating projection algorithm, and is given by

$$\mathbf{x}_{n+1} = P_B P_A \mathbf{x}_n. \tag{1.60}$$

If both constraint sets are convex, the error is reduced at each projection, so the ER algorithm is guaranteed to converge, although convergence may be slow. This is known as *projection onto convex sets*. The ER algorithm is likely to stagnate for non-convex constraints. A diagram showing the behaviour of the ER algorithm for convex and non-convex constraint sets is shown in Fig. 1.6.

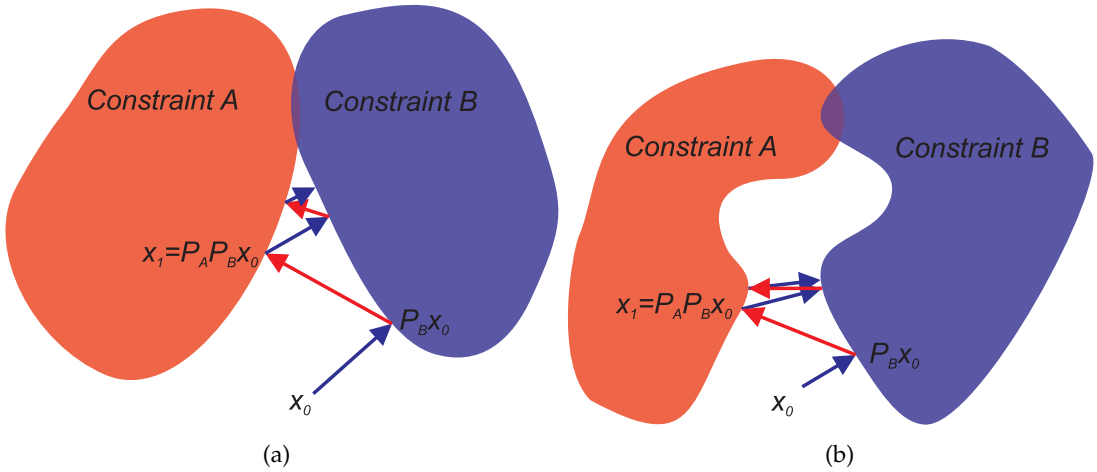


Figure 1.6 Behaviour of the ER algorithm for (a) convex constraints and (b) non-convex constraints.

1.3.3.2 Relaxed projection algorithm

The relaxed projection (RP) algorithm is simply alternating relaxed projections. The relaxation parameters are usually chosen such that $0 < \gamma < 1$, and the relaxed projections are used in place of P_A and P_B in Eq. (1.60) so that

$$\mathbf{x}_{n+1} = T_B T_A \mathbf{x}_n. \tag{1.61}$$

The RP algorithm is sometimes more effective than the ER algorithm at avoiding stagnation, but its ability to avoid stagnation in general is quite limited. A diagram showing the behaviour of the RP algorithm for non-convex constraints is shown in Fig. 1.7(a). Note that the best final estimate of the solution is not an iterate $\mathbf{x}$, but is one of the projections from the iterate, $P\mathbf{x}$.

The RP algorithm is commonly described as the "solvent flipping" algorithm [12] when applied to a support/solvent constraint with the relaxation parameter γ_I for the support projection P_I set to 1, and the Fourier magnitude relaxation parameter set to $\gamma_F = 0$, where the constraints and projections are as described in Sec. 1.3.4. The "charge flipping" algorithm is similar to solvent flipping [13].

1.3.3.3 Douglas-Rachford algorithm

The Douglas-Rachford (DR) algorithm [14] is given by the iteration

$$\mathbf{x}_{n+1} = \frac{1}{2}(R_B R_A \mathbf{x}_n + \mathbf{x}_n) \tag{1.62}$$

$$= \mathbf{x}_n + P_B R_A \mathbf{x}_n - P_A \mathbf{x}_n. \tag{1.63}$$

The Douglas-Rachford algorithm was originally defined for convex constraints where convergence is guaranteed since if both constraints are convex, $R_A R_B \mathbf{x}_n$ is closer to the so-

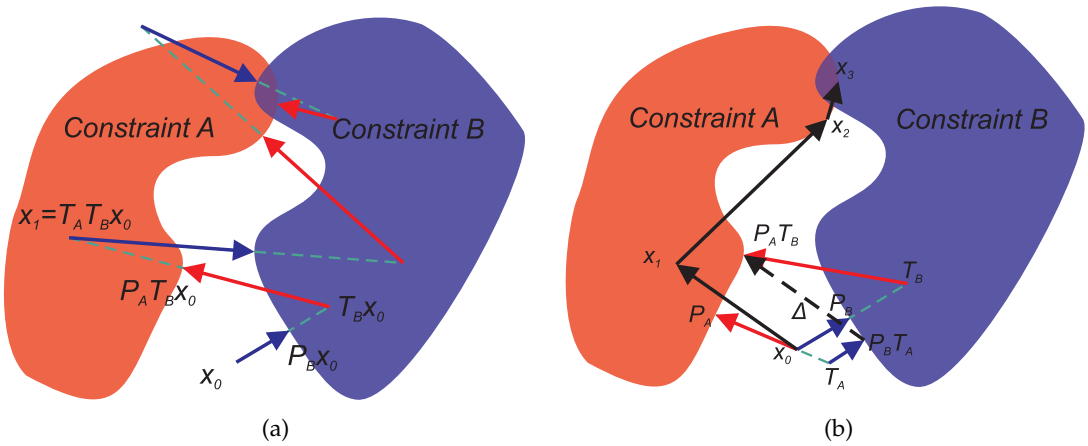


Figure 1.7 Behaviour of the RP and DM algorithms for non-convex constraints (a) RP algorithm and (b) DM algorithm.

lution than $\mathbf{x}_n$, and therefore so will $\mathbf{x}_{n+1}$ since it is the average of $R_AR_B\mathbf{x}_n$ and $\mathbf{x}_n$. The DR algorithm performs remarkably well for non-convex constraints and like many great ideas, variants of the DR algorithm have been discovered independently in many disparate fields [15, 16, 17].

1.3.3.4 Hybrid input-output algorithm

A very popular and perhaps the predecessor to all modern iterative projection algorithms is the hybrid input-output (HIO) algorithm [15]. The hybrid input-output (HIO) algorithm was defined specifically for support and Fourier magnitude constraints in image reconstruction and is given by

$$\mathbf{x}_{n+1}[\mathbf{t}] = \begin{cases} g[\mathbf{t}] & \text{if } \mathbf{t} \in S \\ \mathbf{x}[\mathbf{t}] - \beta g[\mathbf{t}] & \text{if } \mathbf{t} \notin S, \end{cases} \tag{1.64}$$

where $\mathbf{g} = P_M\mathbf{x}_n$ is the Fourier magnitude projection, S is the support, and β is a parameter usually set between 0.7 and 1.

Various attempts have been made to extend the HIO algorithm to general constraints, with all of the algorithms presented later in this chapter at least in part inspired by the HIO algorithm. The HIO algorithm is equivalent to the difference map algorithm described later when P_A is a Fourier magnitude constraint and P_B is a support and/or positivity constraint with $\gamma_A = \beta^{-1}$ and $\gamma_B = -1$.

1.3.3.5 Generalized hybrid input-output algorithm

The HIO was described in terms of support, positivity, and Fourier magnitude constraints only. Millane [18] extended the idea of the HIO algorithm to the case of more general image

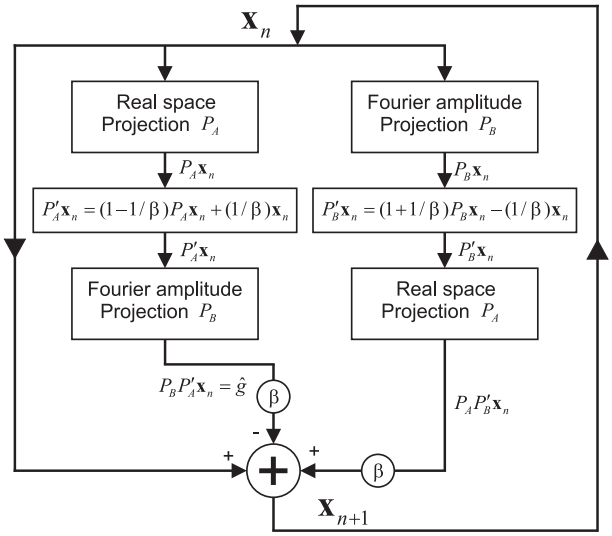


Figure 1.8 Block diagram of the DM algorithm.

domain constraints as

$$\mathbf{x}_{n+1}[\mathbf{t}] = \begin{cases} g[\mathbf{t}] & \text{if } |c[\mathbf{t}] - g[\mathbf{t}]| \leq \epsilon \\ \mathbf{x}_n[\mathbf{t}] + \beta(c[\mathbf{t}] - g[\mathbf{t}]) & \text{if } |c[\mathbf{t}] - g[\mathbf{t}]| > \epsilon, \end{cases} \tag{1.65}$$

where ϵ is a small tolerance factor that allows the algorithm to terminate when close to a solution within the noise level of the data, $\mathbf{c} = P_I\mathbf{g}$ is the result of the image domain constraints applied to the output of the Fourier magnitude projection. This can be written in terms of projections as

$$\mathbf{x}_{n+1}[\mathbf{t}] = \begin{cases} P_M\mathbf{x}_n[\mathbf{t}] & \text{if } |P_IP_M\mathbf{x}_n[\mathbf{t}] - P_M\mathbf{x}_n[\mathbf{t}]| \leq \epsilon \\ \mathbf{x}_n[\mathbf{t}] + \beta(P_IP_M\mathbf{x}_n[\mathbf{t}] - P_M\mathbf{x}_n[\mathbf{t}]) & \text{if } |P_IP_M\mathbf{x}_n[\mathbf{t}] - P_M\mathbf{x}_n[\mathbf{t}]| > \epsilon. \end{cases} \tag{1.66}$$

1.3.3.6 Difference map algorithm

The difference map (DM) algorithm is defined by the iteration [16]

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \beta[P_AT_B\mathbf{x} - P_BT_A\mathbf{x}], \tag{1.67}$$

where $\beta \neq 0$ is a parameter usually set such that $0.7 \leq |\beta| \leq 1$, and the relaxation parameters γ_A and γ_B are generally set to $-1/\beta$ and $1/\beta$, respectively. Note that a negative β is equivalent to interchanging the constraints A and B. A block diagram of the DM algorithm is shown in Fig. 1.8.

If the DM algorithm stagnates, i.e. $\mathbf{x}_{n+1} = \mathbf{x}_n$, then $P_A T_B \mathbf{x} = P_B T_A \mathbf{x} = \hat{\mathbf{x}}$. Since $\hat{\mathbf{x}}$ is the result of both projections P_A and P_B, $\hat{\mathbf{x}}$ must satisfy both constraints. So the DM does not stagnate unless it is at a solution. It can be shown that it moves towards, and then away from, a point closest to both constraint sets, and continues to explore the parameter space [16]. It is therefore quite effective at avoiding stagnation. Furthermore, the iterates $\mathbf{x}$ are not themselves the estimate of the solutions, which are given by $P_A T_B \mathbf{x}$ or $P_B T_A \mathbf{x}$ instead. Usually, the more restrictive projection is used as the estimate, or often the image domain projection is preferred since the final solution usually is presented or used in the image domain. The requirement to use a projection to estimate the solution is not unusual, and applies to all the other algorithms described in this chapter except for the ER algorithm.

Unless $|\beta| = 1$, the DM algorithm uses twice as many projections per iteration compared to the ER, RP and DR algorithms. If $\beta = 1$, $\gamma_A = -1/$ and $\gamma_B = 1$, then the DM algorithm is identical to the Douglas-Rachford algorithm. The DM algorithm is shown diagrammatically in Fig. 1.7(b).

1.3.3.7 Hybrid projection reflection algorithm

The hybrid projection reflection (HPR) algorithm [17] is given by

$$\mathbf{x}_{n+1} = \frac{1}{2}[R_A(R_B + (\beta - 1)P_B) + \mathbf{x} + (1 - \beta)P_B], \tag{1.68}$$

The HPR takes longer than the HIO to converge to the vicinity of the solution, but provides consistently better solutions with more stability when near the solution.

1.3.3.8 Relaxed averaged alternating reflectors algorithm

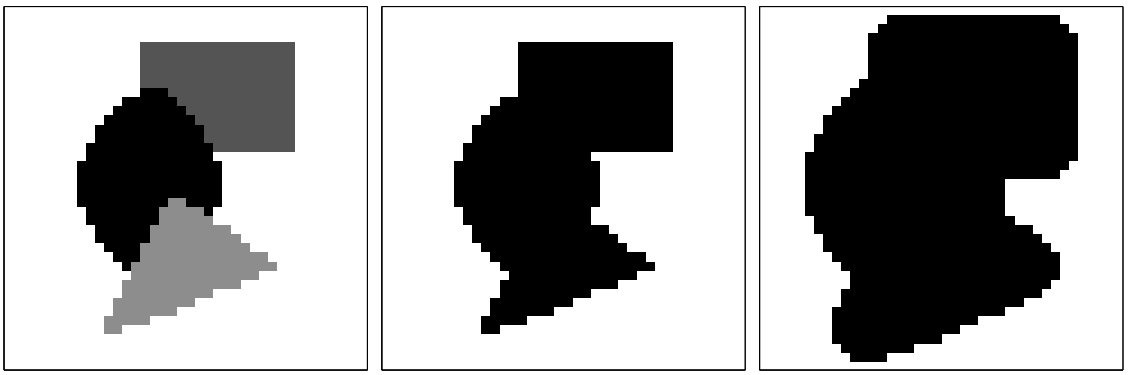
The relaxed averaged alternating reflectors (RAAR) algorithm [19] is given by

$$\mathbf{x}_{n+1} = \frac{1}{2}\alpha(R_A R_B + \mathbf{x}) + (1 - \alpha)P_B. \tag{1.69}$$

The RAAR algorithm was originally defined using a Fourier magnitude constraint for constraint B and a support and positivity constraint for constraint A. It is designed to retain the performance of the HIO algorithm while converging to the vicinity of the solution while providing the improved quality and stability of the HPR algorithm when refining the final solution. The RAAR algorithm is a concatenation of the DR algorithm and a second step which takes a relaxed projection of the iterate onto the constraint set B. This provides additional stability with noisy data, and is further discussed in Sec. 2.10.

1.3.4 Common crystallographic constraint sets and their projections

Some common crystallographic constraint spaces and their projections are now described.



(a) (b) (c)

Figure 1.9 Support constraint. (a) Original image, (b) support, (c) loose support.

1.3.4.1 Support and known value constraint

It is often known where an object or image is located, i.e. where it is non-zero. This region is called the "support" of the image. A support constraint is defined by the set

$$\{\mathbf{x} : x[\mathbf{t}] = 0, \quad \mathbf{t} \notin S\}, \tag{1.70}$$

where S is the support of the image. The support constraint manifold is a hyperplane of dimension $2N - 2|S|$ in $\mathbb{R}^{2N}$, where $|S|$ is the cardinality of set S. The support constraint is therefore a convex constraint of infinite extent. Fig. 1.9 shows an image and its support, with a loose support also depicted. Note that choosing S to be too large weakens the constraint, while choosing S to be too small causes the image to no longer satisfy the constraint.

The support constraint occurs in many image reconstruction fields such as crystallography, astronomy, and MRI. An example of the use of a support constraint was described in Sec. 1.2, where a diffraction experiment is conducted on a single object, as opposed to a crystal, and the Fourier magnitudes are finely sampled. The fine sampling of the Fourier magnitudes corresponds to zero-padding in the image domain, which is exploited with a support constraint.

A related constraint is knowledge that the object is real, that is to say, the imaginary parts of all pixel values are zero, i.e.

$$\{\mathbf{x} : Im[x[\mathbf{t}]] = 0, \quad \forall \mathbf{t}\}. \tag{1.71}$$

The real image constraint set is a hyperplane of dimension N in $\mathbb{R}^{2N}$.

In some cases the values outside the support are a constant value ρ instead of 0, so the

constraint becomes

$$\{\mathbf{x} : x[\mathbf{t}] = \rho, \quad \mathbf{t} \notin S\}, \tag{1.72}$$

which is called a constant value constraint. The geometry of the constraint manifold in $\mathbb{R}^{2N}$ is a hyperplane orthogonal to the axes and is therefore a shifted, but otherwise identical, set to that of a support constraint.

The known value constraint is related to the support constraint, and occurs when the values at a set of pixels are known, that is

$$\{\mathbf{x} : x[\mathbf{t}] = \rho_{\mathbf{t}} \quad \mathbf{t} \notin S\}, \tag{1.73}$$

where $\rho_{\mathbf{t}}$ denotes the known value of the pixel at position $\mathbf{t}$. The geometry of the manifold is a hyperplane orthogonal to the axes and is therefore a shifted, but otherwise identical, manifold to that of a support constraint. The known value constraint is found in holography, where a known reference object is imaged along with the true object.

The projection P_{sup} of $\mathbf{x}$ onto the support (or known value) constraint set is applied by setting all known pixels of $\mathbf{x}$ to their known values. That is,

$$P_{sup}x[\mathbf{t}] = \begin{cases} \rho_{\mathbf{t}} & \mathbf{t} \notin S \\ x[\mathbf{t}] & \mathbf{t} \in S \end{cases} \tag{1.74}$$

In some cases the value of a constant value constraint is not known, i.e. ρ in Eq. (1.72) is unknown. This is called the *floating support constraint* and is given by

$$\{\mathbf{x} : x[\mathbf{t}] = x[\mathbf{t}'], \quad \mathbf{t}, \mathbf{t}' \notin S\}. \tag{1.75}$$

This constraint manifold is a hyperplane of dimension $2N - 2|S| + 1$ in $\mathbb{R}^{2N}$, and is given by collapsing all dimensions corresponding to the pixels in S to the line represented by Eq. 1.75. The floating support constraint is a convex constraint of infinite extent, and its projection P_{fsup}, denoted is applied by setting the values outside S to the average of the values outside S, that is,

$$P_{fsup}x[\mathbf{t}] = \begin{cases} \frac{1}{N-|S|} \sum\limits_{\mathbf{t}' \notin S} x[\mathbf{t}'] & \mathbf{t} \notin S \\ x[\mathbf{t}] & \mathbf{t} \in S. \end{cases} \tag{1.76}$$

1.3.4.2 Positivity constraint

In many applications, it is known that a real image is non-negative. This can be written as (assuming the image is real)

$$\{\mathbf{x} : x[\mathbf{t}] > 0, \quad \forall \mathbf{t}\}. \tag{1.77}$$

The constraint manifold is the all-positive "quadrant" of $\mathbb{R}^{2N}$ bounded by the axes. This constraint set is convex and of finite extent, and reduces the volume of the search space of

$\mathbb{R}^{2N}$ by the factor 2^N.

The projection P_{pos} of $\mathbf{x}$ onto the positivity constraint set is applied by setting all pixels with negative values to 0, that is,

$$P_{pos}x[\mathbf{t}] = \begin{cases} x[\mathbf{t}] & x[\mathbf{t}] \geq 0 \\ 0 & x[\mathbf{t}]t < 0 \end{cases} \tag{1.78}$$

1.3.4.3 Binary constraint

The binary constraint occurs when each pixel of an image is known to be one of two values, usually 0 or 1. This can be written as

$$\{\mathbf{x} : x[\mathbf{t}] \in \{0, 1\}, \quad \forall \mathbf{t}\}. \tag{1.79}$$

The proportion of pixels equal to 1 is denoted f and is referred to here as the *fill fraction* f. If f is known, then in addition to Eq. 1.79, there is the additional constraint

$$\sum_{\mathbf{t}} x[\mathbf{t}] = fN. \tag{1.80}$$

or

$$||\mathbf{x}|| = fN \tag{1.81}$$

The proportion of all possible binary images of size N with fill fraction f, $B(f, N)$, is

$$B(f, N) = \frac{^N C_{fN}}{2^N}. \tag{1.82}$$

For any N, $B(f, N)$ is maximized when $f = 0.5$. $B(f, N)$ is a measure of the "size" of the constraint set and can be directly associated with the "strength" of the constraint. A plot of $B(f, 100)$ is shown in Fig. 1.10(a). Since $B(f, N) = B(1 - f, N)$, the strength of the binary constraint is symmetrical around $f = 0.5$. Furthermore, the values of 0 and 1 can be any values and the nature of the constraint is unchanged. Reversing 0 and 1 to invert the image changes the fill fraction from f to $1 - f$.

The binomial distribution of Eq. (1.82) can be approximated with a Gaussian of mean 0.5 and variance $0.25/N$, i.e.

$$B(f, N) \approx \frac{1}{\sqrt{0.5\pi N}} e^{-2N(f-0.5)^2}. \tag{1.83}$$

As N increases, distribution gets narrower. So the strength of the binary constraint (defined as the proportion of binary images that have a fill fraction within, say, 0.02 of the true fill fraction) for f close to 0.5 decreases, but the strength of the binary constraint for f further from 0.5 increases.

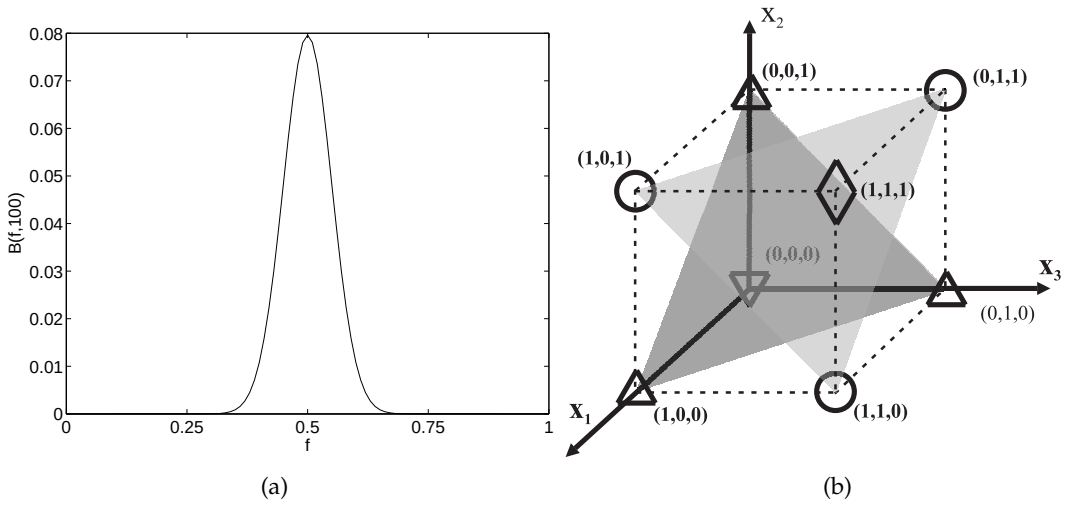


Figure 1.10 Binary and fill fraction constraint. (a) The strength of the binary fill fraction constraint $B(f, 100)$ vs f. (b) Binary and Fill fraction constraints in Euclidean space for $N = 3$. The binary constraint is the points marked by the four symbols. The binary and fill fraction constraints are the points marked by the symbols $\nabla(fN = 0)$, $\triangle(fN = 1)$, $\circ(fN = 2)$, and $\diamond(fN = 3)$.

The binary constraint is a zero-dimensional constraint consisting of the corners of a hypercube as shown in Fig. 1.10(b) for a 3-pixel image. The fill fraction then selects out the corners which are on the $N-1$ dimensional hyperplane orthogonal to the $(1, 1, .., 1)$ vector if applied to a non-binary image, i.e. in the form $\sum x[\mathbf{t}] = fN$. This is true for both $(0, 1)$ sets and general two-valued sets. The binary constraint is therefore highly non-convex, potentially making convergence of IPAs difficult. However, due to its zero-dimensionality, it is a strong (restrictive) constraint and is highly noise resistant.

The binary constraint is a special case of the more general *histogram constraint*, where the number of pixels p_k of value k is known. The histogram constraint is defined by

$$\{\mathbf{x} : n(k) = p_k, \quad \forall k\} \tag{1.84}$$

where $n(k)$ denotes the number of pixels in $\mathbf{x}$ with value k. Like the binary constraint, the histogram constraint is a zero-dimensional constraint. The number of images which satisfy a histogram constraint is

$$H(\mathbf{p}, N) = \frac{N!}{\prod_k p_k!}, \tag{1.85}$$

where $\mathbf{p} = (p_1, p_2, ...)$.

The binary projection, denoted P_{Bin}, is given by

$$P_{Bin}x[\mathbf{t}] = \begin{cases} 0 & x[\mathbf{t}] < 0.5 \\ 1 & x[\mathbf{t}] \geq 0.5. \end{cases} \tag{1.86}$$

The binary with fill fraction projection is given by

$$P_{BF}x[\mathbf{t}] = \begin{cases} 0 & x[\mathbf{t}] \notin S(f) \\ 1 & x[\mathbf{t}] \in S(f), \end{cases} \tag{1.87}$$

where $S(f)$ is the set of the fN largest values of $\mathbf{x}$. The simplest way to find $S(f)$ is to sort the values of the image and choose the largest fN values. In general, the time taken by the sort function is of order $N \log N$, which is reasonably fast, especially when compared to the considerably slower connectivity projection discussed below. If necessary, a faster method would be to recognize that since each image is related to the image at the previous iteration, a good estimate of the cutoff value can be found from the cutoff value of the previous iteration. Only values near the previous cutoff value need to be sorted, while the rest can be directly assigned.

The histogram projection, denoted P_{hist}, is given by

$$P_{hist}x[\mathbf{t}] = \vec{p}(O\{x[\mathbf{t}]\}) \tag{1.88}$$

where $O\{x[\mathbf{t}]\}$ denotes the index of the value $x[\mathbf{t}]$ when $\mathbf{x}$ is sorted and $\vec{p}$ is a sorted N pixel vector with p_k pixels of value k for all k.

1.3.4.4 Atomicity constraint

Some images are made up of small isolated regions and are zero elsewhere. This is like a support constraint with small separated supports of unknown location. An example is a high resolution image of a molecule, where each of the individual atoms are distinct. The atomicity constraint is mathematically difficult to define, but the classic equation of Sayre [20] enforces the atomicity constraint by "blurring" the square of the image, i.e.

$$P_{atom}\mathbf{x} = \mathbf{x}^2 \otimes \mathbf{g} \tag{1.89}$$

where $\mathbf{g}$ is the point spread function chosen depending on the size and shape of each "atom". Simple application of Eq. (1.89) as a projection generally fails, but more sophisticated variations have been shown to work [21].

1.3.4.5 Fourier value constraint

The Fourier constraint applies when the complex values of the Fourier transform are known at a subset of points in Fourier space, that is

$$\{\mathbf{x} : X[\mathbf{h}] = \varrho[\mathbf{h}], \quad \mathbf{h} \in W\}, \tag{1.90}$$

where $\varrho[\mathbf{h}]$ are the known complex values of the Fourier transform at positions $\mathbf{h} \in W$. If all the values are known ($|W| = 2N$), the constraint set reduces to a single point and the solution is immediately obtained. The Fourier value constraint is a known value constraint

in Fourier space and is therefore a convex constraint of infinite extent. One of the earliest applications for iterative algorithms was the Gerchberg-Saxton algorithm [11], which used an ER algorithm to successfully combine the Fourier value at low resolution and support constraints to extend the resolution of images.

The Fourier value projection P_F is given by

$$P_F\mathbf{x} = F^{-1}[\tilde{P}_F F[\mathbf{x}]], \tag{1.91}$$

where $\tilde{P}_F$ is the corresponding constraint in Fourier space, i.e.

$$\tilde{P}_F X[\mathbf{h}] = \begin{cases} \varrho[\mathbf{h}], & \mathbf{h} \in W \\ X[\mathbf{h}], & \mathbf{h} \notin W. \end{cases} \tag{1.92}$$

1.3.4.6 Fourier magnitude constraint

As described in Sec. 1.2, it is often not possible to measure the phase of the diffraction pattern, so only (some or all of) the magnitudes of the Fourier transform are known. The constraint set is given by

$$\{\mathbf{x} : |X[\mathbf{h}]| = M[\mathbf{h}], \quad \mathbf{h} \in W\}, \tag{1.93}$$

where $M[\mathbf{h}]$ denotes the known value of the Fourier magnitude at $\mathbf{h}$ at the set of points W. The Fourier magnitude constraint is the intersection of a set of $|W|$, $(2N-2)$-dimensional hypercylinders in $\mathbb{R}^{2N}$ and is a non-convex constraint. Fig. 1.11 depicts one of the hypercylinders in a 3-D space.

The Fourier magnitude projection P_M is given by

$$P_M\mathbf{x} = \mathcal{F}^{-1}[\tilde{P_M}\mathcal{F}[\mathbf{x}]], \tag{1.94}$$

where $\tilde{P}_M$ is the corresponding constraint in Fourier space, i.e.

$$\tilde{P}_M X[\mathbf{h}] = \begin{cases} M[\mathbf{h}]e^{i\angle X[\mathbf{h}]} & \mathbf{h} \in W \\ X[\mathbf{h}] & \mathbf{h} \notin W \end{cases} \tag{1.95}$$

where $\angle X[\mathbf{h}]$ is the phase of $X[\mathbf{h}]$.

Various more complicated Fourier magnitude projections are possible. For example, when there is noise in the measurements, one could project onto an annulus instead of the fixed magnitude.

1.3.4.7 Symmetry

If the image contains identical "subunits", then it contains *symmetry* as described in Sec. 1.2.3. Symmetry in an image can be defined in terms of *symmetry operations*. A *symmetry opera-*

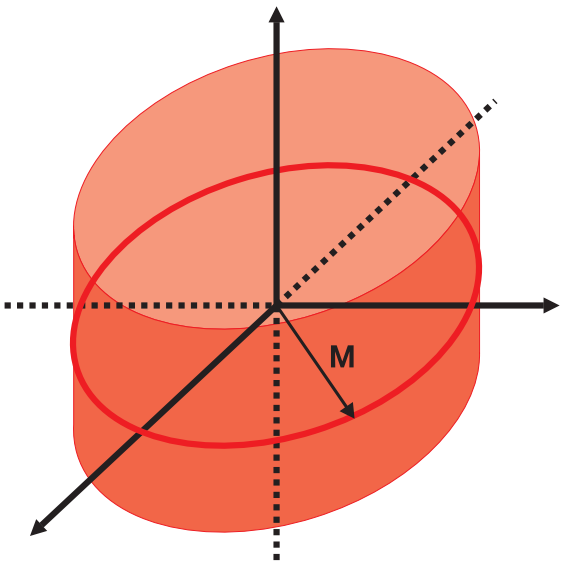


Figure 1.11 The Fourier magnitude constraint in a 3-D Euclidean space.

tion on an image maps the repeated subunits onto each other and is a distance preserving transformation consisting of a combination of rotation, reflection, and translation. An image which *satisfies* a symmetry operation is an image which is invariant when the symmetry operation is applied to it. Since the image is invariant under the symmetry operation, then the image must also be invariant under repeated applications of the symmetry operation. The full set of thus related symmetry operations is called a *symmetry group*.

Choose one of the symmetry operations in the symmetry group C and denote it C_1. Then use repeated applications of C_1 to generate the rest of the group, i.e.

$$C_k = C_1^k, \tag{1.96}$$

where C_1^k denotes k applications of the operator C_1. If C is a *complete* symmetry, then after some minimum of q symmetry operations the cycle repeats itself irrespective of which symmetry operation in the set was chosen, i.e.

$$C_q = I, \tag{1.97}$$

where I is the identity operator. The complete symmetry C is therefore a group in the mathematical sense and q is known as the *order* of the group or symmetry. Note that if q has two factors $q = q_1 q_2$, then the symmetry operations consisting of $C_{q_2}^k$ will also form a group of order q_1 and vice versa. If C is an *incomplete* symmetry, then the cycle does not repeat itself.

The symmetry projection corresponds to symmetry averaging, i.e. the value at an im-

age sample point is set equal to the average over all the symmetry related sample points. Two cases arise depending on whether or not all the points that are symmetry-related to a sample point are sample points themselves. If they are, the non-interpolated symmetry projection P_{sym} is given by

$$P_{sym}x[\mathbf{t}] = \begin{cases} \frac{1}{q}\sum_{k=1}^{q} x[C_k\mathbf{t}] & \text{for } \mathbf{t} \in Q \\ x[\mathbf{t}] & \text{for } \mathbf{t} \notin Q, \end{cases} \tag{1.98}$$

where $C_k\mathbf{t}$ denotes the new position of $\mathbf{t}$ after the symmetry operation C_k is applied and Q is the region over which the symmetry applies.

If the symmetry-related points are not all sample points, then the values of the image at the symmetry-related points must be estimated from the values at neighbouring sample points. The symmetry projection P_{isym} is then given by

$$P_{isym}x[\mathbf{t}] = \begin{cases} \frac{1}{q}\sum_{k=1}^{q} x'(C_k\mathbf{t}) & \text{for } \mathbf{t} \in Q \\ x[\mathbf{t}] & \text{for } \mathbf{t} \notin Q, \end{cases} \tag{1.99}$$

where $x'(C_k\mathbf{t})$ is the interpolated value of $\mathbf{x}$ at point $C_k\mathbf{t}$. The symmetry constraint and its projections are described in more detail in Chapters 5 and 6.

1.4 Error metrics

As with any iterative algorithm, it is important to be able to assess progress, or convergence, of the algorithm and to be able to decide when an acceptable solution has been found. Error metrics are used for this purpose. For an IPA, the definition and choice of error metrics is non-trivial. Various error metrics are used when analyzing the performance of image reconstruction algorithms. For the case of Fourier imaging, they can broadly be divided into either image or Fourier domain metrics. With a projection algorithm, the final estimate is generally obtained as a projection onto the constraints in one of the domains, which will invalidate some of the metrics. For example, the Fourier magnitude error metric will always return 0 if the Fourier magnitude projection is used.

This thesis is primarily concerned with reconstructing electron densities from the Fourier magnitudes, so the image domain representation of the object is more important. The estimate of the answer is therefore usually a projection onto the image domain. Furthermore, the image domain metrics are the most useful, but they require the true solution to be known, and are therefore not usually available. The error metrics are therefore divided based on whether or not the true solution needs to be known for the error metric to be calculated. Lists of common error metrics of both types are shown in the next section.

Converging to an image which satisfies all the constraints means that the algorithm will

stagnate and the error metrics which measure distance from the constraints return zero. Then there are three possibilities.

First, the algorithm has found the true solution, and all useful error metrics will return a value of zero.

Second, the algorithm has found an image which is *trivially related* to the true solution in the sense that it is not the true solution but can be easily modified to reconstruct the true solution. For example, if the constraints cannot tell an image and its circularly shifted variants apart, the circularly shifted variants are *trivially related* to the true solution. An ideal error metric should return zero error if a trivially related solution is found. An example is for a phase retrieval problem with a binary and Fourier magnitude constraint. The circularly shifted and inverted versions of the solution have the same Fourier magnitudes, and are therefore trivially related to the solution.

Third, the algorithm may have found a *false* solution, i.e. a solution which satisfies the constraints but is not related to the true solution in any way, meaning that the constraints are insufficient to uniquely define the solution. Since it is usually not possible to prove that the constraints uniquely define the solution, a simple method to ensure that the solution found is the true solution is to use a *free* error metric, where some of the data is withheld from the algorithm and used after convergence to test the veracity of the solution [22, 23]. Note that the metrics which can only be calculated if the original image is known are generally able to distinguish the true solution from a false solution.

Calculation of some of the error metrics is computationally expensive. This is especially true for those that give a non-zero metric for trivially-related solutions, since the estimated solution needs to be compared to all possible trivially related solutions.

With non-convex problems, the iterate does not move continuously towards the solution, and may even leave the vicinity of the solution after convergence due to noise. One of the metrics can then be used as an estimate of the error, and the iterate with the lowest error metric is used as the final estimate.

Another use for metrics is in detecting stagnation. If an algorithm stagnates or is stuck in a limit cycle, the error metrics will also stagnate or enter a limit cycle, which can be detected and algorithm halted or restarted.

1.4.1 A catalog of error metrics

In the metrics below, $\mathbf{x}$ and $\hat{\mathbf{x}}$ denote the correct and estimated image respectively, and $\mathbf{X} = \mathcal{F}\{\mathbf{x}\}$ and $\hat{\mathbf{X}} = \mathcal{F}\{\hat{\mathbf{x}}\}$ are their respective Fourier transforms. Unless otherwise mentioned, the summations are over all pixels in the image for both real and imaginary values.

1.4.1.1 Metrics which do not require the true solution to be known

The summations in this section are over all measured Fourier magnitudes. The RMS Fourier error is given by

$$\text{RMS Fourier Magnitude Error} = \sqrt{\frac{\sum_{\mathbf{h}}(|\hat{X}[\mathbf{h}]| - |X[\mathbf{h}]|)^2}{\sum_{\mathbf{h}}|X[\mathbf{h}]|^2}}. \tag{1.100}$$

The normalized Fourier magnitude error is the most commonly used crystallographic metric and is usually known as the R-factor. It is given by

$$\text{R-factor} = \frac{\sum_{\mathbf{h}}||\hat{X}[\mathbf{h}]| - |X[\mathbf{h}]||}{\sum_{\mathbf{h}}|X[\mathbf{h}]|}. \tag{1.101}$$

In general an R-factor of 0.2 indicates reasonably good convergence in the electron density determination part of x-ray crystallography. If only some of the Fourier magnitudes are known, then the summations are over the known Fourier magnitudes. If the zero-frequency Fourier magnitude value is not known, then these metrics are image domain level-shift invariant.

The free R-factor is another common metric [22, 23]. In this metric, a few Fourier magnitude values are withheld from the algorithm during refinement. The reconstructed Fourier magnitudes are then compared with the measured Fourier magnitudes. This method has a chance to detect if an algorithm has found a false solution due to insufficient constraints.

The Fourier magnitude correlation coefficient is given by

$$\text{Fourier Magnitude Correlation Coefficient} = \frac{\sum_{\mathbf{h}}(|X[\mathbf{h}]| - \overline{|X|}) \cdot (|\hat{X}[\mathbf{h}]| - \overline{|\hat{X}|})}{\sqrt{\sum_{\mathbf{h}}(|X[\mathbf{h}]| - \overline{|X|})^2 \sum_{\mathbf{h}}(|\hat{X}[\mathbf{h}]| - \overline{|\hat{X}|})^2}}, \tag{1.102}$$

where $\overline{|X|}$ and $\overline{|\hat{X}|}$ denote the mean Fourier magnitudes of the true and estimated images $\mathbf{x}$ and $\hat{\mathbf{x}}$ respectively. The Fourier magnitude correlation coefficient is scale and level shift invariant, and returns a number between -1 and 1, with 0 being no correlation and 1 indicating a perfect match to the data.

The Sigma-A (σ_A) error is commonly used for probabilistic algorithms and is given by [2]

$$\sigma_A = \sqrt{\frac{\sum_{\mathbf{h}}[(|\hat{X}[\mathbf{h}]|^2 - \overline{|\hat{X}|}^2) \cdot (|X[\mathbf{h}]|^2 - |\bar{X}|^2)]}{\sqrt{(\sum_{\mathbf{h}}[|X[\mathbf{h}]|^2 - |X[\bar{\mathbf{h}}]|^2])(\sum_{\mathbf{h}}[|\hat{X}[\mathbf{h}]|^2 - \overline{|\hat{X}|^2}])}}}. \tag{1.103}$$

Another type of metric which does not require the true solution to be known is to determine the absolute or normalized error between the two projections in a projection algorithm. Alternatively, the change between successive iterates can be used as a metric, since a change of 0 corresponds to stagnation. The RMS value of the term $\delta = P_A R_B \mathbf{x} - P_B R_A \mathbf{x}$ in the DM equation in Eq. (1.67) is an example of such an error metric.

1.4.1.2 Metrics which require knowledge of the true solution

The root mean square (RMS) error in the image is the distance in Euclidean space between the estimate and the solution and is given by

$$\text{RMS Image Error} = \sqrt{\frac{\sum_{\mathbf{t}} |\hat{x}[\mathbf{t}] - x[\mathbf{t}]|^2}{\sum_{\mathbf{t}} |x[\mathbf{t}]|^2}}. \tag{1.104}$$

Due to the distance preserving property of the Fourier transform, the RMS Fourier error is identical to the RMS Image error, where

$$\text{RMS Fourier Error} = \sqrt{\frac{\sum_{\mathbf{h}} |\hat{X}[\mathbf{h}] - X[\mathbf{h}]|^2}{\sum_{\mathbf{h}} |X[\mathbf{h}]|^2}} \tag{1.105}$$

$$= \text{RMS Image Error}. \tag{1.106}$$

The numerator is the distance between the estimated and correct image in Euclidean space, and the denominator is a normalization factor. In principle the unnormalized error is more useful in analyzing projection algorithms since projection algorithms are Euclidean origin-independent, i.e. if the starting point and constraints are shifted by a vector $\mathbf{d}$ in Euclidean space, then all resulting projections and iterates are identical but shifted by a vector $\mathbf{d}$ also.

The image correlation coefficient is given by

$$\text{Image Correlation Coefficient} = \frac{\sum_{\mathbf{t}} (x[\mathbf{t}] - \bar{x}) \cdot (\hat{x}[\mathbf{t}] - \bar{x})}{\sqrt{\sum_{\mathbf{t}} (x[\mathbf{t}] - \bar{x})^2 \sum_{\mathbf{t}} (\hat{x}[\mathbf{t}] - \bar{\hat{x}})^2}}. \tag{1.107}$$

where $\bar{x}$ and $\bar{\hat{x}}$ denotes the mean values of $\mathbf{x}$ and $\hat{\mathbf{x}}$ respectively.

The weighted phase error is scale and level shift invariant in the image domain and is given by

$$\text{Phase Error} = \frac{\sum_{\mathbf{h}} |X[\mathbf{h}]| \cdot |\angle X[\mathbf{h}] - \angle \hat{X}[\mathbf{h}]|}{\sum_{\mathbf{h}} |X[\mathbf{h}]|}, \tag{1.108}$$

being careful to put the phase difference on the interval $(-\pi, \pi)$. The weighted phase error is between 0 and π, with a value of 0 indicating perfect correlation, π indicating perfect

correlation with the negative of the image, and $\pi/2$ indicating no correlation with the data. In general a phase error of about $\pi/6$ is reasonably good.

The weighted cosine phase error is scale and level shift invariant in the image domain and is given by

$$\text{Cosine Phase Error} = \sum_{\mathbf{h}} \frac{(X[\mathbf{h}] \cos(\angle \hat{X}[\mathbf{h}] - \angle X[\mathbf{h}])}{|X[\mathbf{h}]|}. \tag{1.109}$$

The cosine phase error returns a number from 1 to -1, with 1 indicating perfect correlation, -1 indicating perfect correlation with the negative of the image, and 0 indicating no correlation with the data. The unweighted (by $X|[\mathbf{h}]|$) cosine phase and phase errors can also be calculated.

Chapter 2

Aspects of Projection Algorithms

2.1 Introduction

Some aspects of projection algorithms are presented in this chapter. Projection algorithms of non-convex sets by their nature are chaotic and very difficult to describe analytically. However, their behaviour is not completely unpredictable, and it is possible to make some heuristic observations about them. The purpose of this chapter is to attempt to provide some insight into the approximate behaviour of these algorithm by means of some simple examples. The observations which follow in this chapter are based on the author's experiences with these algorithms and should not be thought of as absolute truths.

A model of the convergence of projection algorithms is presented in Sec. 2.2, along with examples and an application of the model. This is followed by a study of the behaviour of the algorithm when in the convergence region near the solution. This is followed by short discussions on the desirable properties of IPAs, the effects of uniqueness and false solutions, methods for handling more than two constraints, bootstrapping, estimating the solution and the effects of noise and possible methods for improving the performance of algorithms for very noisy data. Finally, representing projection algorithms as a map is discussed.

2.2 Progress model of a projection algorithm

Projection algorithms come in many different types, but other than for the ER and RP algorithms, the projection algorithms described in Sec. 1.3.3 are all variants of a Difference map type scheme. For these algorithms, it is proposed that their progress can be broadly be divided into three phases. The phases are described below, and an example of the convergence behaviour for a particular algorithm is presented. Then studies of the distribution of the number of iterations needed for convergence are shown which corroborates the progress model.

(A) Finding the attractor The first phase is where the iterate goes from its random start onto an *attractor* [24]. An attractor is a term used in the study of dynamical systems, and is used to describe a manifold in Euclidean space which draws the iterates towards it [25]. Once in the attractor, the iterates generally do not leave the attractor. In many problems, there is only one attractor but some problems have more than one. If the iterate enters the wrong attractor, i.e. one which does not contain the solution, it will never converge, or perhaps take an extremely large number of iterations to leave the attractor. Thus the choice of initialization can be critical to the success of an algorithm. The number of iterations taken to enter an attractor is a small number, and for simplicity is modeled as a constant number of iterations T_{C1}. Furthermore, from each starting point the algorithm may enter a number of attractors, only one of which contains the true solution, so there is a probability p that the algorithm has entered the correct attractor and will eventually converge.

(B) Finding the convergence region of the solution The algorithm wanders in the attractor until it enters the *convergence region* of the solution. The convergence region is defined in the next paragraph. Near solutions will also have a convergence region, but if the distance between constraints is large enough, the iterate will move away. Similarly if the distance between constraints at the true solution is large, the fixed point may not be attractive enough and the algorithm will move away. If there are no near-solutions which have a distance between constraints of similar size to the true solution, the algorithm will wander the attractor until it enters the convergence region of the true solution. A model for the number of iterations required in this phase is that there is a constant probability μ at each iteration of the iterate entering the convergence region of the true solution [26, 24]. Therefore, the number of iterations I required for the algorithm to find the convergence region can be modeled by a geometric distribution $\omega(1-\omega)^{I-1}$, where the parameter ω is strongly influenced by the size of the attractor and the size of the convergence region of the solution

(C) Converging to the solution Once the algorithm has entered the convergence region of the solution, it converges near-monotonically at a rate determined by the distance and effective "angle" between the constraints. The effective "angle" is discussed in Sec. 2.3.2. The number of iterations required for convergence is modeled as a constant number T_{C2}, which is shown to be a reasonably good model in Sec. 2.2.7.

2.2.1 A diagram of the convergence sets in Euclidean space

A diagram of the various convergence sets in the Euclidean space is shown in Fig. 2.1. Note that many near-solutions also have regions of attraction where the iterate spends more time, but in general the iterate will soon move away from these false convergence regions.

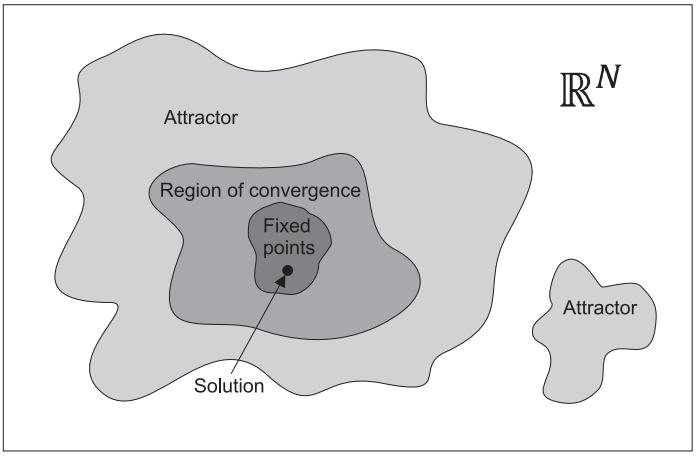


Figure 2.1 Set diagram showing the relationships between the attractors, convergence regions, fixed point sets, and solution in $\mathbb{R}^{2N}$.

2.2.2 An example of projection algorithm behaviour

A particular example is now used to analyze the convergence behaviour of a projection algorithm. The example problem consists of binary and Fourier magnitude constraints for a 128×128 pixel binary image, and is referred to as Problem #1. The binary image is shown in Fig. 2.2. The DM algorithm is used with $\beta = -0.7$. The zero-dimensional nature of the binary constraint means that convergence is achieved quickly and precisely once in the convergence region, but more iterations may be required in the attractor phase. A typical error plot is shown in Fig. 2.3(a) where the initial, search, and convergence phases can be clearly seen. The error plots for five runs of the same problem with different starting points are shown in Fig. 2.3(b). It can be seen that the initial and convergence phases take approximately the same number of iterations for all five runs, and the three phases can be clearly distinguished using only the error metric plot.

Using the run from Fig. 2.3(a), snapshots of the iterate $\mathbf{x}$ at various stages in the run are shown in Fig. 2.4. Two images of $\mathbf{x}$ before the algorithm has converged to the attractor are shown in Fig. 2.4(a) and Fig. 2.4(b). It can be seen that the iterate does not resemble the solution in any way. Images of the iterate during the search of the attractor are shown in Fig. 2.4(c) and Fig. 2.4(d). Various candidate solutions are being tried, and the images somewhat resemble the true solution shown in Fig. 2.4(h) in the sense that they are of approximately the same size and have a similar degree of compactness. By serendipity, one of the shapes tried during the search phase at iteration 45 sufficiently resembles that of the true solution and the algorithm now enters the convergence region of the true solution. The image of the iterate at the point at which the algorithm first enters the convergence region of the solution is shown in Fig. 2.4(e). Small corrections are now made to the image as shown in Fig. 2.4(f), which shows the iterate midway through the convergence phase. The final iterate which is at a fixed point is shown in Fig. 2.4(g), and the final estimate shown in Fig. 2.4(h). The final estimate is a perfect reconstruction and is given by one of

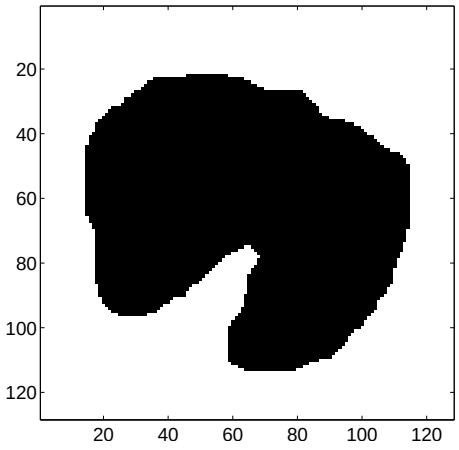


Figure 2.2 The 128×128 pixel binary image to be reconstructed.

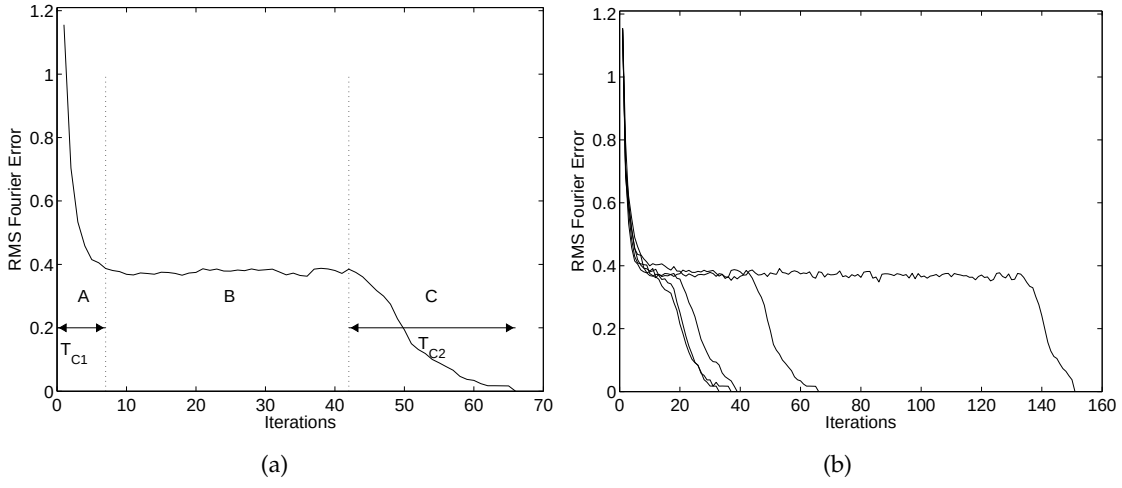
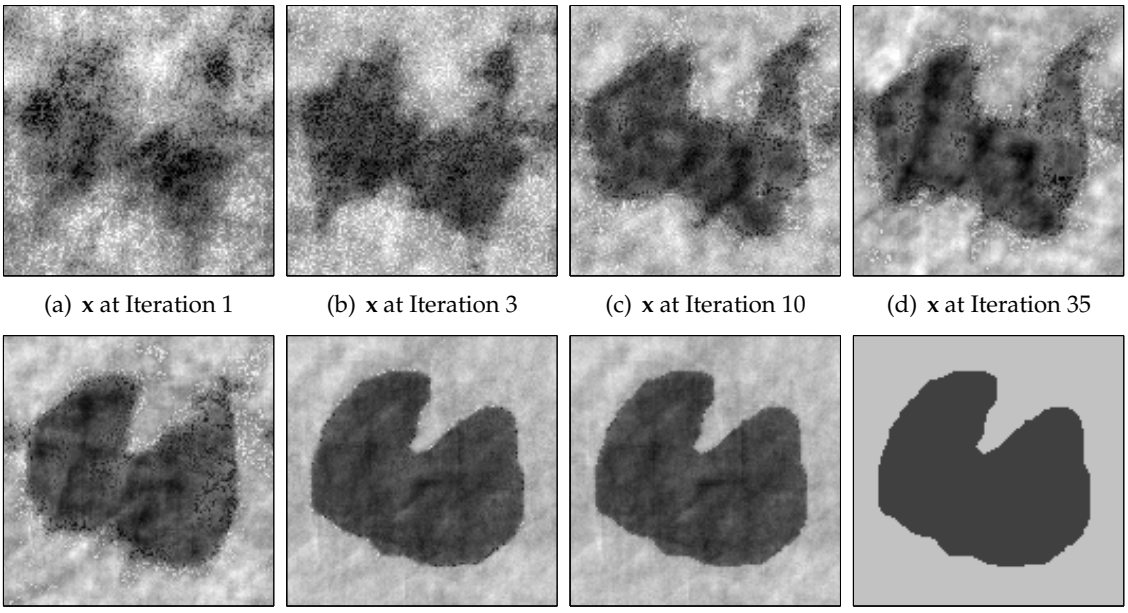


Figure 2.3 (a) Error vs Iteration for one run (b) Error vs Iteration for 5 different starting points



(a) **x** at Iteration 1 (b) **x** at Iteration 3 (c) **x** at Iteration 10 (d) **x** at Iteration 35

(e) **x** at Iteration 45 (f) **x** at Iteration 55 (g) **x** at Iteration 66 (h) $P_B T_M \mathbf{x}$ at Iteration 66

Figure 2.4 The image **x** at various stages of convergence. (a) **x** at Iteration 1 (b) **x** at Iteration 3, converging towards the attractor (c) **x** at Iteration 10, on the attractor (d) **x** at Iteration 35, on the attractor (e) **x** at Iteration 45, just entering the convergence phase (f) **x** at Iteration 55, in the convergence phase (g) **x** at Iteration 66, at a fixed point (h) the final estimate $P_B T_M \mathbf{x}$ at Iteration 66 is the correct solution.

the projections onto the binary constraint.

2.2.3 The relationship between the constraints and the convergence behaviour

The effect of small constraint sets is to reduce the number of candidate solutions, so the attractor manifold should be smaller, and it is likely that fewer iterations are needed to find the convergence region. For problems where the constraint sets are either zero-dimensional or highly non-convex, the convergence region has been observed to generally be very small. A possible explanation could be because little information about the global geometry of the constraint sets in Euclidean space can be inferred from the local behaviour of the projections, i.e. the constraint set manifolds are poorly linearized by the linear projection operators. The result of a small convergence region is that the search phase lasts a long time.

For problems where the constraint sets are convex or non-discrete, the main difficulty may be in convergence with "small angle" problems as described in Sec. 2.3.2, so the final convergence phase may take relatively more time. This is illustrated by considering a problem where the highly non-convex binary constraint in Problem #1 is replaced by the convex support constraint (Problem #2). The same DM algorithm is used and the error plots are

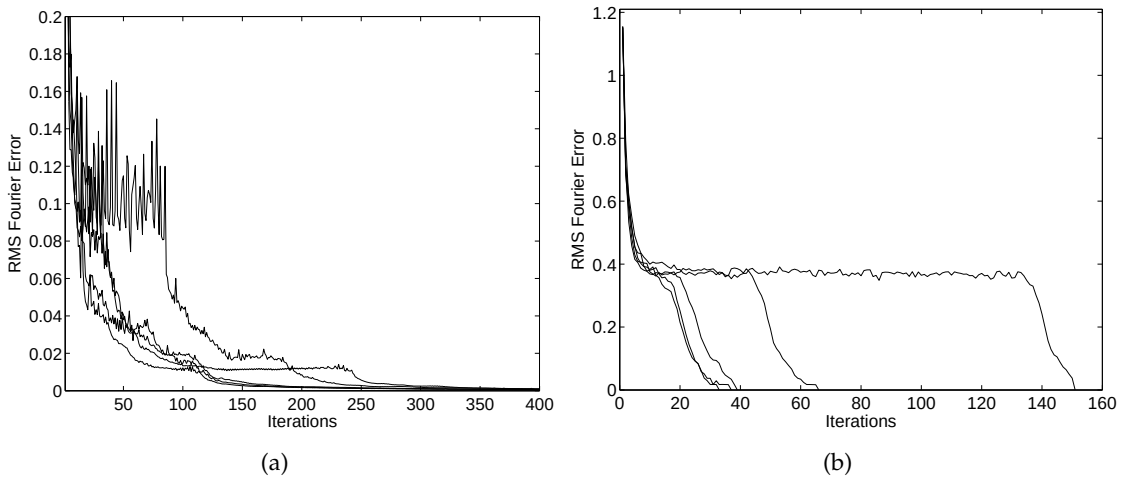


Figure 2.5 Error metric vs Iteration for five runs for a (a) convex support constraint (b) zero-dimensional non-convex binary constraint.

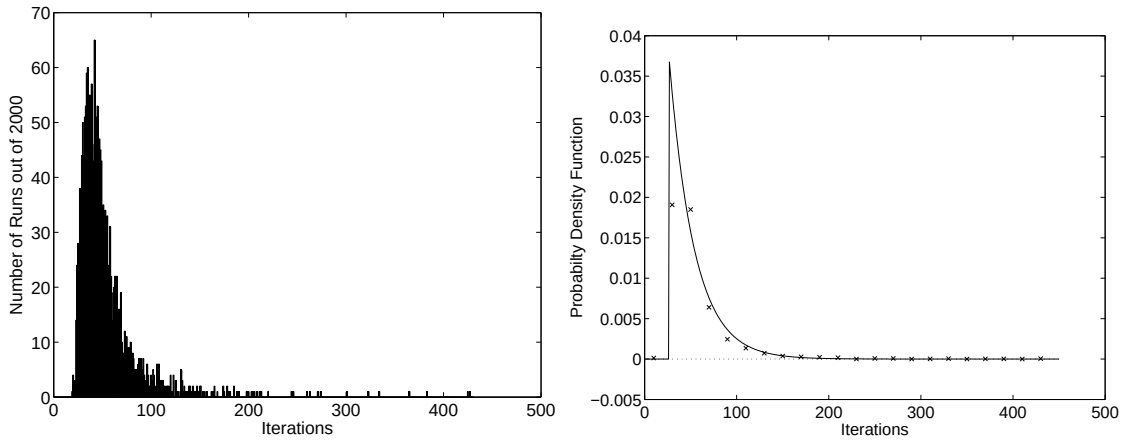
compared to those for the binary constraint case in Fig. 2.5. For the convex support constraint, many iterations are required in the final phase of converging to the correct solution. For the binary problem, the convergence phase is relatively short compared to the amount of time spent searching the attractor.

2.2.4 Model for the number of iterations required

A possible model for the number of iterations needed for successful convergence of an algorithm is now described. The algorithm is run until either the solution is found or the maximum I_{max} number of iterations is reached. Note that the correct solution is not found for iterations which reach I_{max}. A binary constraint is used so convergence is defined as when the image is perfectly reconstructed. As described earlier, there is a probability p that the algorithm enters the correct attractor and will eventually converge. The attractor search phase is modeled as a constant probability process, with a probability ω at each iteration that the algorithm enters the convergence region. The number of iterations required is therefore a geometric distribution $\omega(1-\omega)^{I-1}$, which can be approximated by an exponential distribution $\lambda e^{-\lambda I}$ with $\lambda = -ln(1-\omega)$. The initial convergence to the attractor and convergence phase are modeled as requiring a constant number of iterations T_{C1} and T_{C2} respectively. The probability density function (pdf) for the number of iterations I at convergence is then given by

$$P(I) \quad = \quad \begin{cases} 0 & \text{if } I < T_C \text{ or } I > I_{max} \\ p\lambda e^{-\lambda(I-T_C)} & \text{if } T_C \leq I < I_{max} \\ (1-p) + pe^{-\lambda(I_{max}-T_C)} & \text{if } I = I_{max} \end{cases} \tag{2.1}$$

where $T_C = T_{C1} + T_{C2}$ is the sum of the iterations required in two constant phases of convergence. Note that if I_{max} is sufficiently large so that the number of iterations which



(a) (b)

Figure 2.6 (a) Histogram of the number of runs which stop at each iteration for the same problem as in Fig. 2.3. The effect of T_C can clearly be seen. (b) pdf of (a) along with a fitted exponential function using Eq. (2.1) with parameters $p = 1$, $T_C = 27$ and $\lambda = 0.037$.

converge after I_{max} iterations is negligible, then the term $pe^{I_{max}-T_C} = 0$. Then if T_C and p are known, the pdf can easily be modified to create a standard exponential distribution, and λ is then the inverse of the mean of the modified pdf.

The binary problem #1 was run 2000 times and the histogram of the distribution of the number of iterations at convergence is plotted in Fig. 2.6(a). The histogram was normalized to a pdf and fitted to Eq. (2.1) as follows. T_C was estimated to be about 27 iterations, and $p = 1$ since all runs converged. The mean μ of the scaled data $P(I + T_C)/p$ was calculated to be $\mu = 27.2$ and λ was set to $1/\mu$, giving $\lambda = 0.037$. The original pdf along with the fitted function is shown in Fig. 2.6(b).

2.2.5 Restarting the algorithm to optimize the expected number of iterations necessary for convergence

If $p < 1$, then there is a possibility that the algorithm will never converge if the wrong starting point is chosen. This problem can be resolved by restarting the algorithm (reinitializing the iterate) if it does not converge after some fixed number of iterations I_r. If $p = 1$, then the memoryless property of the exponential distribution along with the "fixed cost" T_C means that there is no advantage to be gained in restarting the algorithm. Note that this does not preclude the use of multiple start points in parallel if there is more than one computer available. The expected number of iterations needed for successful convergence as a

function of I_r, $I_s(I_r)$, is then

$$\begin{aligned}
I_s(I_r) &= \frac{\int_0^{I_r} IP(I)dI + I_r(1 - \int_0^{I_r} P(I)dI)}{\int_1^{I_r} P(I)dI} \\
&= \frac{\int_{T_C}^{I_r} p\lambda I e^{-\lambda(I-T_C)}dI + I_r(1 - \int_{T_C}^{I_r} p\lambda e^{-\lambda(I-T_C)}dI)}{\int_{T_C}^{I_r} p\lambda e^{-\lambda(I-T_C)}dI} \\
&= \frac{[-pIe^{-\lambda(I-T_C)} - \frac{p}{T_C}e^{-\lambda(I-T_C)}]_{T_C}^{I_r} + I_r(1 - [-pe^{-\lambda(I-T_C)}]_{T_C}^{I_r})}{[-pe^{-\lambda(I-T_C)}]_{T_C}^{I_r}} \\
&= \frac{p(T_C + 1/\lambda) - p(I_r + 1/\lambda)e^{-\lambda(I_r-T_C)} + I_r(1 - p + pe^{-\lambda(I_r-T_C)})}{p - pe^{-\lambda(I_r-T_C)}} \\
&= \frac{p(T_C + 1/\lambda) - \frac{p}{\lambda}e^{-\lambda(I_r-T_C)} + I_r(1-p)}{p - p\lambda e^{-\lambda(I_r-T_C)}}. \qquad (2.2)
\end{aligned}$$

The optimum number of iterations I_r after which to restart the algorithm, i.e. which minimizes the value of $I_s(I_r)$, is given by

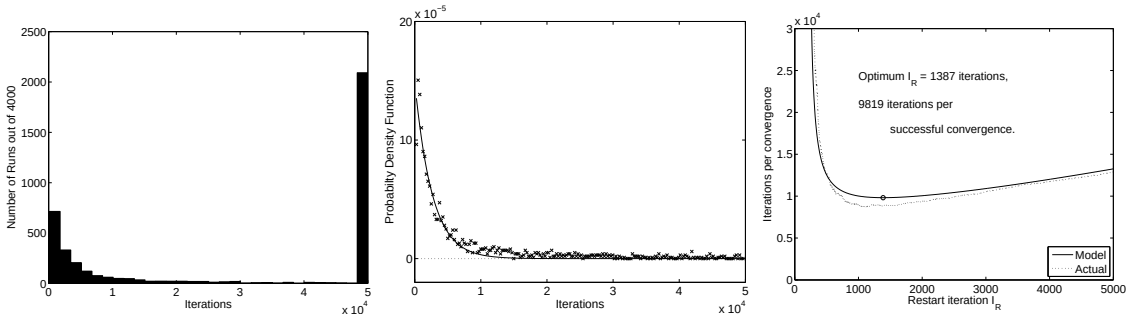
$$\begin{aligned}
\frac{dI_s}{dR} &= 0 \\
(pe^{-\lambda(I_r-T_C)} + 1 - p)(p - p\lambda e^{-\lambda(I_r-T_C)}) &= 0 \\
-(p(T_C + 1/\lambda) - \frac{p}{T_C}e^{-\lambda(I_r-T_C)} + I_r(1-p))(pe^{-\lambda(I_r-T_C)}) &= 0 \\
e^{-\lambda(I_r-T_C)}[2p - 1 - \lambda pT_C - p - I_r\lambda + pI_r\lambda] + 1 - p &= 0, \qquad (2.3)
\end{aligned}$$

which can be solved numerically for I_r.

There is no point in applying the optimization to Problem #1 since $p = 1$. So a third problem (Problem #3) which is identical but with less Fourier magnitude data is run, again with 4000 start points, and the resulting histogram is shown in Fig. 2.7(a). The reduction in data means that more searching is required, but the solution is still unique due to the zero-dimensional binary constraint. The maximum number of iterations was set to $I_{max} = 50000$, so the large peak at the end contains the unconverged runs which were either never going to converge or were going to converge in > 50000 iterations.

The exponential distribution Eq. 2.1 was fitted to the pdf as shown in Fig. 2.7(b) using non-linear least-mean-squares. The parameters were found to be $T_C = 200$, $\lambda = 3.81 \times 10^{-4}$, and $p = 0.364$. Since $p << 1$, it can be inferred that the peak at the end of Fig. 2.7(a) is not just the summation of the runs which would have converged in over 50000 iterations, i.e. some of the runs were never going to converge and have entered the wrong attractor.

The expected number of iterations per successful convergence using both the raw data and substituting the fitted model parameters into Eq. (2.2) is shown in Fig. 2.7(c). Substituting the fitted parameters into Eq. (2.3) and solving gave an optimum $I_r = 1387$ iterations with an expected $I_s = 9819$ iterations.



(a) (b) (c)

Figure 2.7 (a) Histogram of the number of runs which stop at each iteration (b) pdf of (a) with fitted Eq. 2.1 (c) Expected number of iterations per run for each choice of I_R (Eq. 2.2). The optimum $I_R = 452$ corresponding to an expected $I_s = 18118$ iterations per successful convergence is shown.

The choice of optimum value is very sensitive to T_C, which is poorly estimated using the pdf. Equation (2.2) is essentially the first moment of the pdf, and fitting the parameters to it gives a better estimate of the parameters, especially T_C. A least-mean-squares fit of Eq. (2.2) gives the parameters $T_C = 231$, $\lambda = 3.97 \times 10^{-4}$, and $p = 0.375$, resulting in an optimum $I_r = 1483$ iterations with an expected $I_s = 9829$ iterations per successful convergence, which is reasonably close to the values found by fitting the parameters to the pdf.

In practical situations, it is uncommon to know the parameter values λ and p. However, a reset is still desired to prevent the possibility of being trapped in the wrong attractor. It is possible to estimate T_{C1} by starting the algorithm and waiting for the error metric to fall, and T_{C2} by starting the algorithm at some points close to the solution. Then a reasonable choice for I_R could be something like $I_R = 10T_C$. Note that the expected number of iterations per convergence increase more slowly as the choice of I_R increases, so it is better to err on the side of choosing I_R to be too large.

The value of p given by taking the ratio of the converged to total runs is 0.48, which is not the same as the 0.37 given by the model. The pdf has a longer tail than expected, i.e. the model underestimates the number of iterations which take a long time to converge which explains the underestimate of p. Nevertheless, the exponential model gives a good first-order approximation of the behaviour of the algorithms.

A possible explanation for the longer tail is that when the algorithm is searching the space, it moves towards and then away from near-solutions as described previously. So the iterate tends not to return to areas where it has already been, i.e. the algorithm is not completely memoryless, and the probability of convergence increases as the algorithm proceeds, explaining the longer tail. An alternative explanation could be the possibility of multiple attractors with different probabilities of convergence, i.e. different degrees of difficulty in finding the solution which is dependent on the starting position. The final result is then the sum of the exponentials from the attractors, resulting in the longer than expected tail.

2.2.6 Shape of the convergence region

An idea of the shape and size of the convergence region can be gained by starting the algorithm at various locations close to the true solution and counting the number of iterations required for the algorithm to converge to the true solution. If the algorithm is started within the convergence region, convergence should be fast with a monotonically decreasing error.

As an example, the convergence region of a problem with a 128×128 pixel image with binary and Fourier magnitude constraints is mapped. The distribution of the number of iterations needed for perfect convergence from a random start point is shown in Fig. 2.8(a). The mean number of iterations is $1/\lambda \approx 838$. The attractor is therefore large relative to the convergence region since convergence from a random point takes hundreds to thousands of iterations., whereas once in the convergence region convergence typically takes less than 100 iterations. For this problem, any start point which converges in less than $k = 100$ runs was considered to be in the convergence region, since 100 iterations is small compared to $1/\lambda$, but is larger than T_C as is shown later.

Varying amounts of Gaussian, uniform, and binary noise were added to the true solution to generate the starting images and 500 runs were conducted for each noise level and type. The proportion of runs which converge in less than 100 iterations versus the RMS noise level is shown in Fig. 2.8(b). The RMS noise level is defined by

$$\text{RMS Noise Level} = \sqrt{\frac{\sum_{\mathbf{t}} |x_0[\mathbf{t}] - x[\mathbf{t}]|^2}{N}} \tag{2.4}$$

where x_0 and x denote the initial and true images respectively. Using the same data, Fig. 2.8(c) shows the same plot but versus the absolute noise level given by

$$\text{Absolute Noise Level} = \frac{\sum_{\mathbf{t}} |x_0[\mathbf{t}] - x[\mathbf{t}]|}{N}. \tag{2.5}$$

The error in the starting images over which the probability of being in the convergence region falls is well defined with a waterfall region at an RMS error of 1 for this particular problem. The RMS metric performs remarkably well in predicting whether or not the start point is in the convergence region regardless of the distribution of the noise. The absolute error metric has the same waterfall region, but the three curves for the different noise types are misaligned. This suggests that the shape of the convergence region in $\mathbb{R}^{2N}$ is spherical rather than cubical, with a sphere being an excellent first-order estimate of the shape of the convergence region. The radius of the sphere in $\mathbb{R}^{2N}$ is as given by the amount of noise at the waterfall, which is around 1 RMS, corresponding to 128 in the 128^2-dimensional Euclidean space. Note that while the convergence region may be spherical around the solution, the edges of the sphere are likely to be extremely rough. For example, the RMS error when the image is in the attractor search phase is around 0.55 for this problem, so the search of the attractor occurs in one of the indentations of the hypersphere.

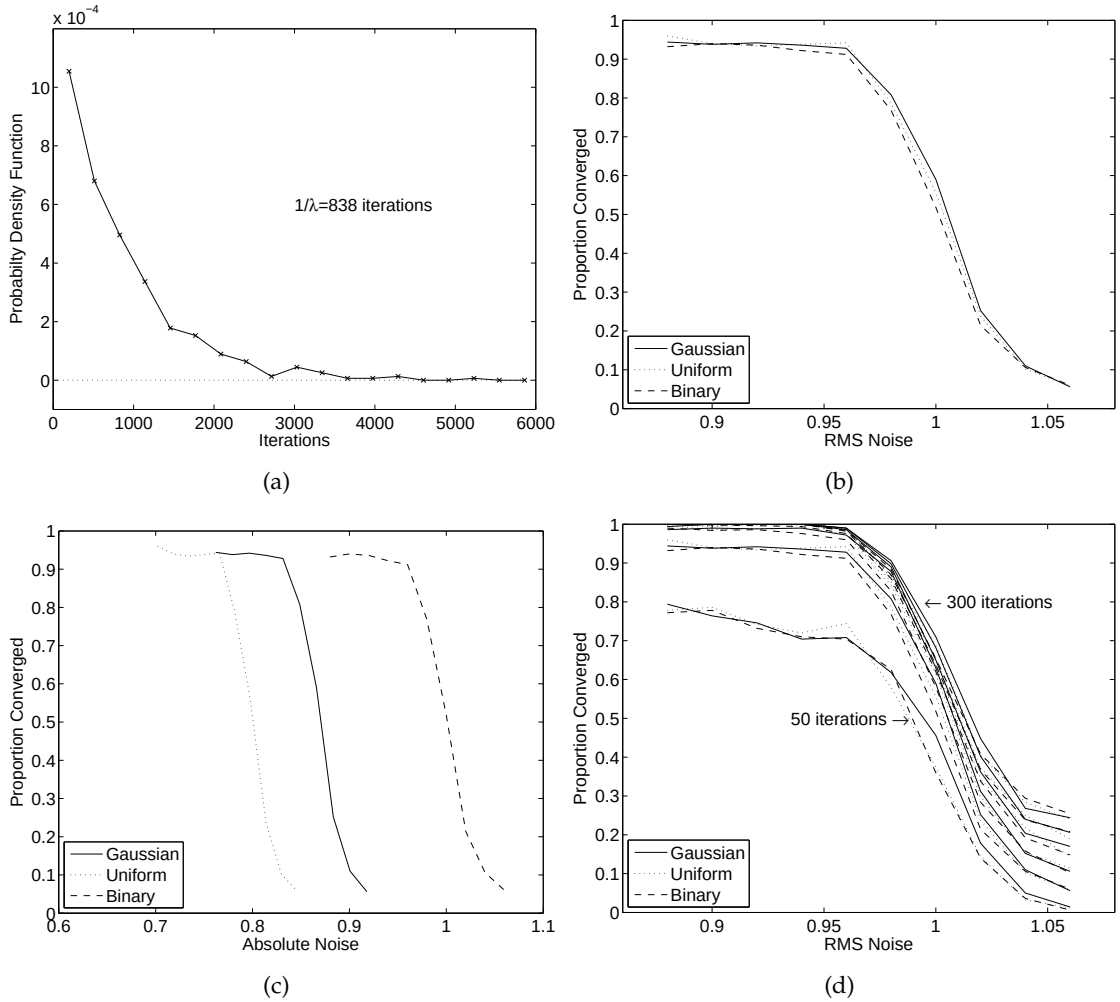
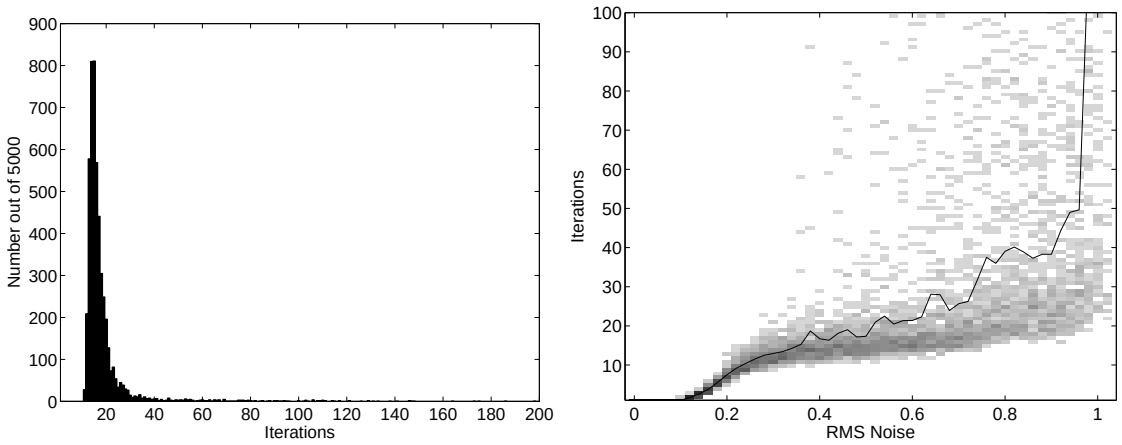


Figure 2.8 (a) Pdf of the number of runs which converge at a given iteration. (b) Proportion of runs which converge in under k=100 iterations vs RMS error. (c) Proportion of runs which converge in under $k = 100$ iterations vs absolute error. (d) Similar plot to (a) but with values of $k = 50, 100, 150, 200, 250, 300$.

Setting k to the values $k = 50, 100, 150, 200, 250, 300$ iterations, a similar plot to that in Fig. 2.8(b) is shown in Fig. 2.8(d). The curves for each distribution are well matched regardless of the choice of k, and the curves for $k > 100$ are all very similar. This suggests that $50 < T_C < 100$, so the minimum choice of 100 iterations is a good choice for k.

2.2.7 Iterations needed for convergence in the convergence region

A small study is now made of the number of iterations needed for convergence when in the convergence region. The same problem as in Sec. 2.2.6 was used and the RMS error of the starting image was set to 0.5. 5000 different starting images were used and run for a maximum of 500 iterations. The number of runs which converge at each number of iterations is shown in Fig. 2.9(a). It can be seen that although there is a long tail, the variance



(a) (b)

Figure 2.9 (a) Number of iterations needed for convergence out of 5000 for a starting image with 0.5 RMS error. (b) Number of iterations needed for convergence for starting images with errors between 0 and 1.1rms. The line represents the mean number of iterations needed for convergence and the grayscale colours show the distribution of the number of runs, with darker colours corresponds to more runs converging in that number of iterations.

overall is reasonably small, especially when compared to the total number of iterations needed for convergence. This suggests that a constant number is an acceptable model for the number of iterations required, especially in the absence of any other information. Furthermore, for some problems such as those with discrete constraints, the number of iterations needed in the convergence region is small, further minimizing any model errors.

The number of iterations needed for convergence is plotted versus the distance (RMS error) from the solution in Fig. 2.9(b). The algorithm converges in 1 iteration when the error is very small (< 0.1RMS). This is because as long as the error of each pixel is less that 0.5, the estimate found by applying the discrete binary projection is the solution. For errors larger than about 1 RMS, the starting iterate is no longer in the convergence region, i.e. the error power is past the waterfall region from Fig. 2.8(b), and so the mean number of iterations needed for convergence rises dramatically. Furthermore, the Fig. 2.9(b) confirms that the choice of 100 iterations as the cutoff for the number of iterations needed in the convergence region was reasonable.

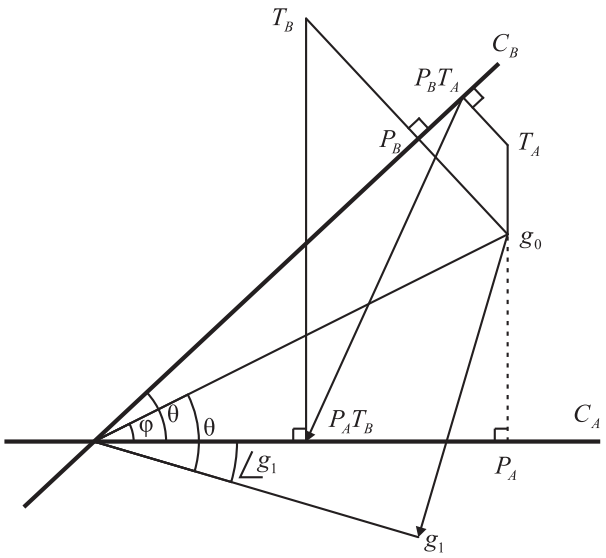


Figure 2.10 DM algorithm for the 2D example.

2.3 Behaviour in the Convergence Region

2.3.1 Convergence for two lines for the DM algorithm

2.3.1.1 A 2-D example

Consider two lines C_A and C_B in 2D space, with each line representing a constraint set. Without loss of generality, let the intersection of the lines be at the origin, and one of the lines to be the x-axis. Denote the angle between the lines by θ. The lines are scale invariant, so the initial position of the iterate is set to be a unit vector $\mathbf{g}_0 = (\cos\phi, \sin\phi)$.

Consider one application of the DM algorithm as illustrated in Fig. 2.10.

The first iterate is at

$$\begin{aligned}\mathbf{g}_1 &= \mathbf{g}_0 + \beta(P_AT_B - P_BT_A) \\ &= (\cos\phi + \beta[(1+\gamma_B)\cos(\theta-\phi)\cos\theta - \gamma_B\cos\phi - \cos^2\theta\cos\beta \\ &\quad -\gamma_A\sin\theta\sin\phi\cos\theta],\ \sin\phi - \beta[\cos\theta\cos\beta\sin\theta - \gamma_A\sin^2\theta\sin\phi]). \qquad (2.6)\end{aligned}$$

Substituting the usual values of $\gamma_A = -1/\beta$ and $\gamma_B = 1/\beta$ into Eq. 2.6, shows that

$$|\mathbf{g}_1| = \cos\theta\sqrt{\cos^2\theta + \beta^2\sin^2\theta} \qquad (2.7)$$

$$\tan(\angle\mathbf{g}_1 - \angle\mathbf{g}_0) = \beta\tan\theta. \qquad (2.8)$$

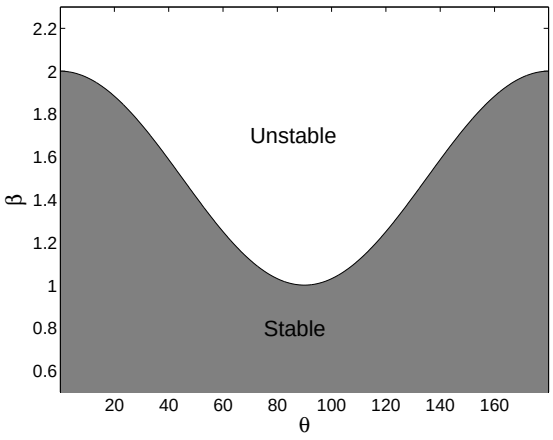


Figure 2.11 Stable values of β for each value of θ using Eq. (2.9).

The algorithm then converges only if

$$\begin{aligned} |\mathbf{g}_1|^2 &< 1 \\ \implies \beta &< \frac{1-\cos^4\theta}{\sin^2\theta}. \end{aligned} \tag{2.9}$$

A plot of Eq. (2.9) is shown in Fig. 2.11. It can be seen that convergence is guaranteed as long as $\beta < 1$, and the algorithm will not converge with $\beta > 2$. The Douglas-Rachford algorithm is thus at the boundary of stability with $\beta = 1$ if $\theta = 90°$.

The value of either component as a function of iteration n is then given by

$$x(n) = \frac{x(0)}{\cos\phi} D^n \cos(\frac{2\pi n}{T} + \phi), \tag{2.10}$$

where $x(0)$ is the initial value of the component, the period T of the sinusoid is given by

$$T = \frac{2\pi}{\tan^{-1}(\beta\tan\theta)} \tag{2.11}$$

iterations, and the damping factor D is

$$D = \cos\theta\sqrt{\cos^2\theta + \beta^2\sin^2\theta}. \tag{2.12}$$

A plot of the positions of the iterate for a 2-pixel problem with $\theta = 15°$ and $\beta = 0.7$ is shown in Fig. 2.12(a). The DM algorithm is seen to spiral towards the solution, with the spiral allowing the algorithm to search the space. Fig. 2.12(b) shows the x-axis value of the iterates, which form a damped sinusoid, the frequency and damping factor of which are determined solely by β and θ.

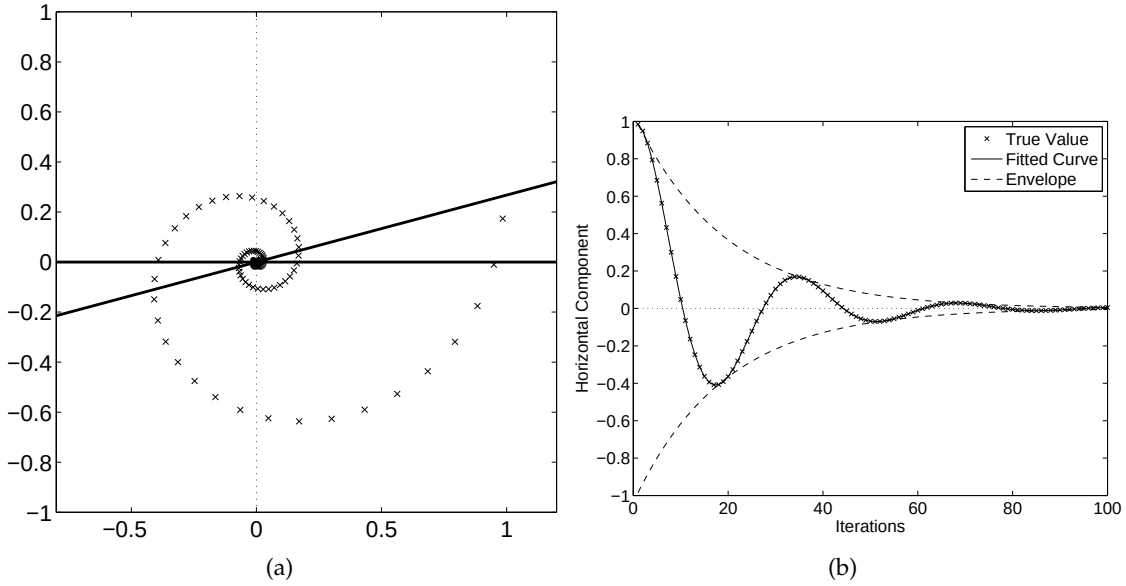


Figure 2.12 2D example for a problem with $\theta = 15°$ using the DM algorithm with $\beta = 0.7$. (a) Positions of iterate and (b) the value of the x-axis versus iterations.

A 2D Euclidean space plot of the next iterate from $\mathbf{g}_0$ for a system with $\theta = 60°$ and β ranging from 0.1 to 1.5 is shown in Fig. 2.13. It can be seen that all the $\mathbf{g}_1$ fall on a line, and the smaller the β, the more direct (and faster) the convergence. The tradeoff is that for more complex problems, the more direct convergence of the smaller β may mean that the algorithm is less able to search the space.

2.3.1.2 A simple 3-D example

The case of two lines is now extended to higher dimensions. Two lines in any dimension can be uniquely identified by 4 points, which in turn identify a 3-D space, so any higher dimensional problem with two single-dimensional linear constraints can be considered in 3-D. Let C_1 and C_2 be two lines in a 3D space. If the lines intersect, then this can be considered as the 2D problem described above. If the lines do not intersect, the iterates spiral towards the intersection in the intersecting dimensions, while moving at a constant rate in the non-intersecting dimension. This is illustrated in Fig. 2.14, which shows a pair of lines defined parametrically by $C_1 = (x, 0, 0)$ and $C_2 = (y \tan(15°), y, 1)$. The iterates spiral in the x and y dimensions while moving at a rate of 1 per iteration in the z direction. The value of the projection of the iterate onto any line (pixel) executes a geometrically damped sinusoid with an added linear term caused by the non-intersecting dimension. However, if the projection is onto a line which is one of the constraints, then there is no drifting linear term.

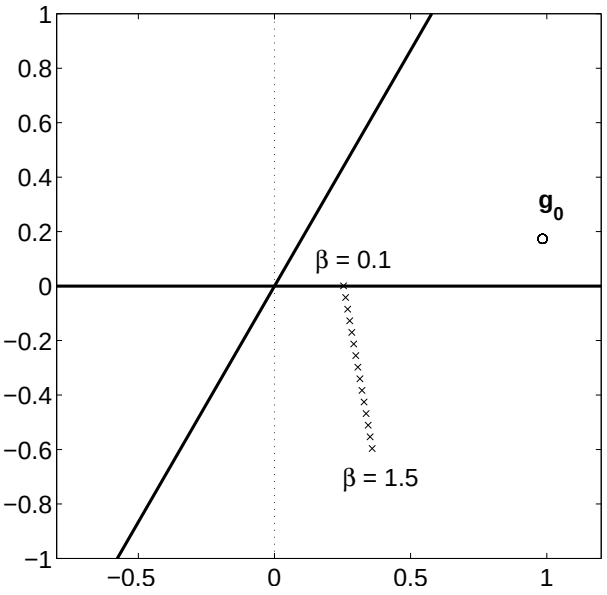


Figure 2.13 2D example for a problem with $\theta = 15°$ showing the position of the next iterate $\mathbf{g}_1$ when using the DM algorithm with β ranging from 0.1 to 1.5.

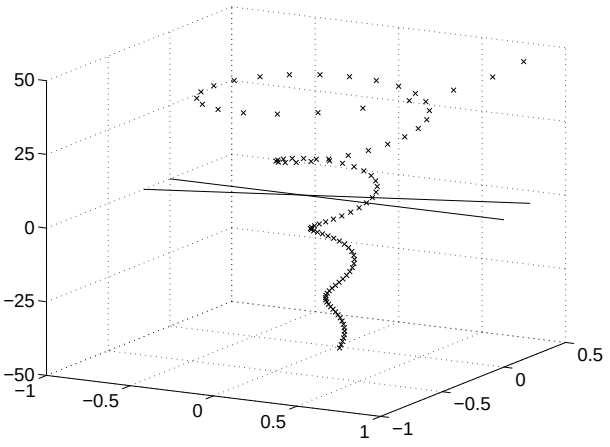


Figure 2.14 Iterates for the simple 3D example with two non-intersecting lines when using the DM algorithm.

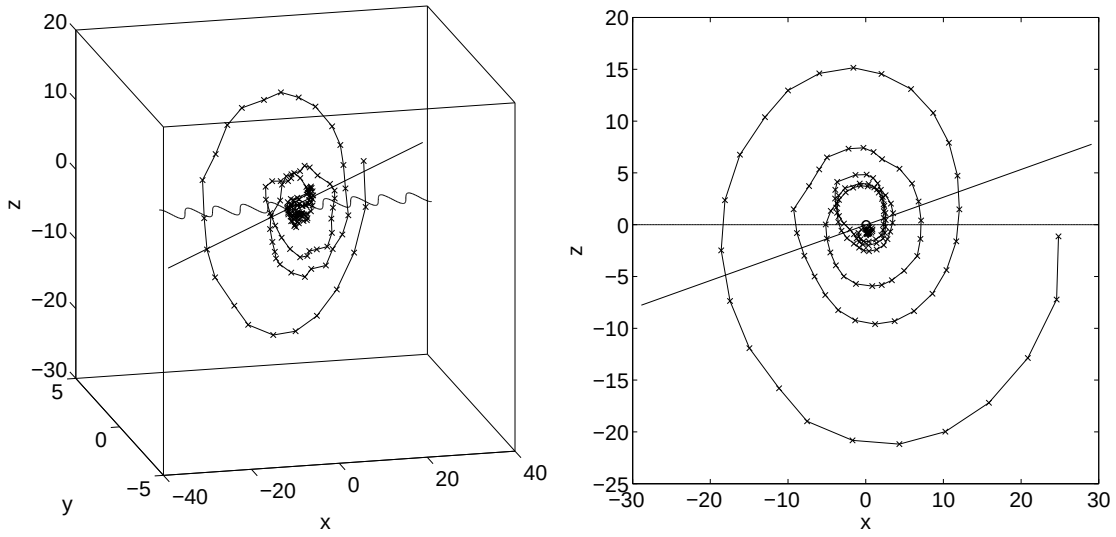


Figure 2.15 Two views (a) 3D view (b) view along the y-axis.

2.3.1.3 A more complex 3-D example

Consider now the non-convex constraint C_1 given by a sinusoidal curve with equation

$$y = M\sin(x) \tag{2.13}$$
$$z = 0 \tag{2.14}$$

where M is the amplitude of the sinusoid, and another convex constraint C_2 consisting of a line inclined at an angle ϕ to the x-axis, with equation

$$z = \tan(\phi)(x+S) \tag{2.15}$$
$$y = \sin(S) \tag{2.16}$$

where S is a shift along the x-axis. The two constraints intersect at $(S, \sin(S), 0)$. With S set to 0, the resulting symmetry causes unusual convergence problems which do not happen for most real-world problems.

The problem converges for a variety of values of ϕ, S and M. The magnitude of the sinusoid was set to $M = 0.5$, $\theta = 15°$, $S = 0.1$ and the DR (DM with $\beta = 1$) algorithm was started from an initial position of $(27, 5, 6)$. A plot of the progression of **g** from two viewpoints is shown in Fig. 2.15.

From Fig. 2.15(b), it can be seen that the usual form of the circular spiralling towards the solution persists, with an effective angle ($\theta_e f f$) of $15.1°$ as shown in Fig. 2.16(a). As the al-

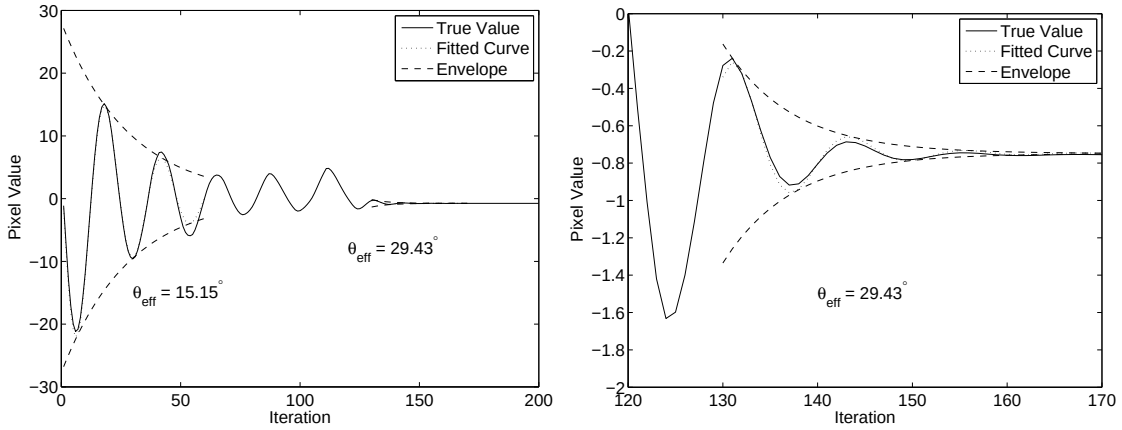


Figure 2.16 Value of the iterate vs Iteration (a) z-Component of the iterate in Fig. 2.15. (b) Zoom in of (a) showing the second geometrically damped sinusoid.

gorithm converges towards the solution, the complexity of the sinusoid geometry disturbs the spiral. When the iterate is very near to the solution, the spiralling form reasserts itself, but with an angle of $29.4°$ as shown in Fig. 2.16(b), which matches well with the local angle between the line and the sinusoid near the solution of $30.1°$.

2.3.2 Convex constraints in higher dimensions

It is much more difficult to define an "angle" in a multidimensional space, but it is still possible to define an "effective angle" θ_{eff}, in the sense that the frequency of the sinusoids and the rate of decay are related in the same way.

The behaviour of the DM algorithm in higher dimensions is studied by reconstructing a 40×40 pixel image with convex support and Fourier value constraints using a DM algorithm with $\beta = 0.7$. The absolute value of a single value of the Fourier transform is plotted versus iteration, and a decaying sinusoid is fitted which assumes a constant "angle", i.e. one parameter which governs both the decay rate and the frequency of the sinusoid. The angle changes during the iterations in a similar manner to that of Fig. 2.16(b), as shown in Fig. 2.17, where a single Fourier value which when fitted to a decaying sinusoid for the first 500 iterations gives $\theta_{eff} = 6.92°$ (Fig. 2.17(b)), but gives $\theta_{eff} = 1.05°$ when fitted to iterations 500 to 2500. Furthermore, a different Fourier value can have a different behaviour as shown in Fig. 2.17(d), which does not exhibit the larger θ in the early iterations.

2.3.2.1 Constrained and unconstrained pixels

The behaviour of pixel values appears to be strongly dependent on whether or not there exists direct information about that pixel in the constraints. The pixel values versus iterations for two pixels are shown in Fig. 2.18. The pixel shown in Fig. 2.18(a) is inside the

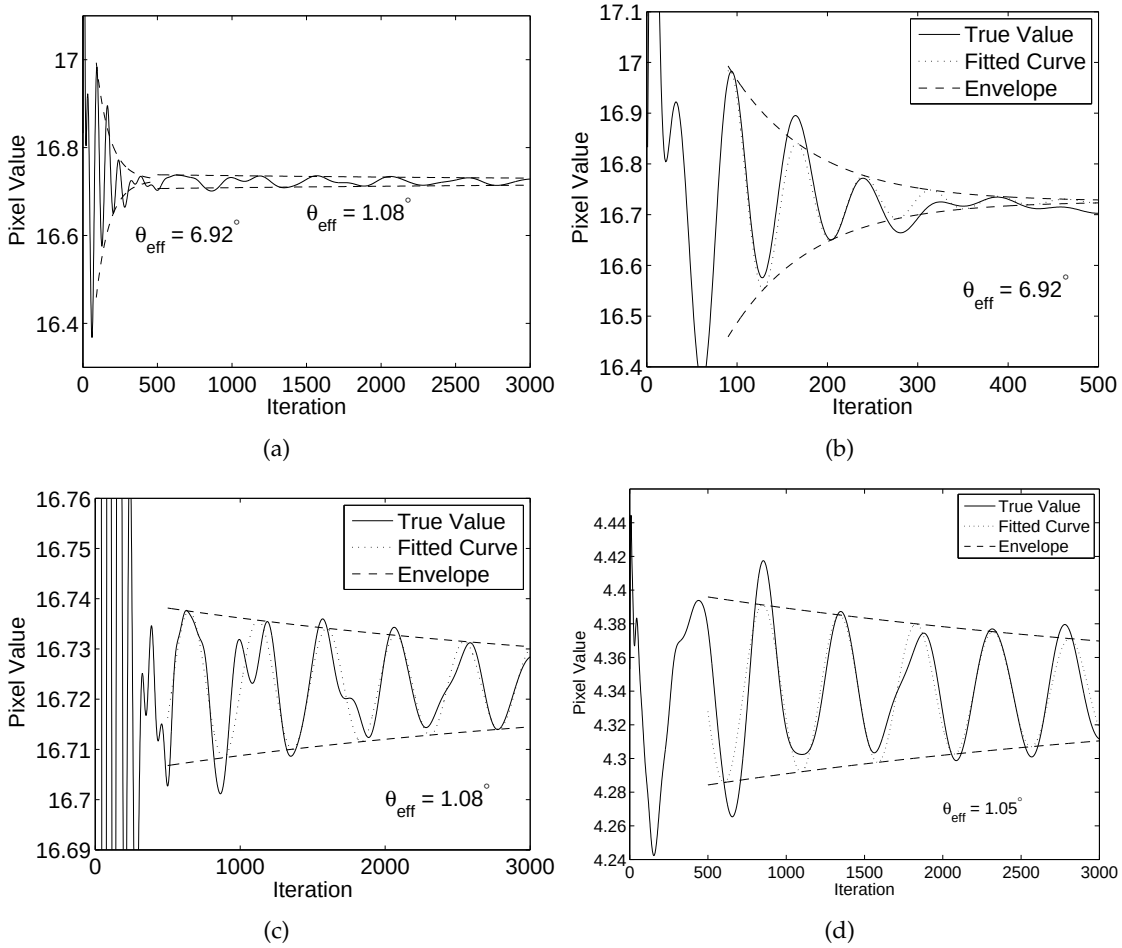
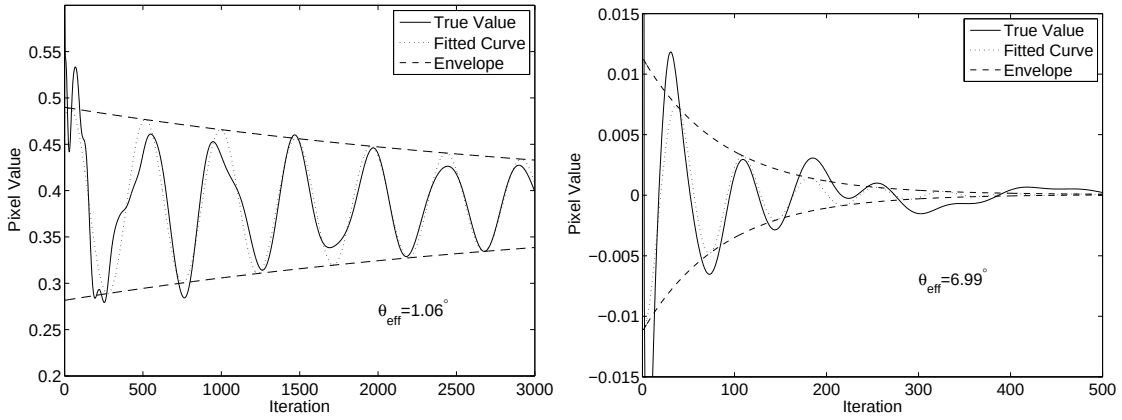


Figure 2.17 Value versus iterations for a Fourier value in a 40×40 pixel image using the DM algorithm with $\beta = 0.7$. (a) All iterations. (b) Zoom in and decaying sinusoidal fit for the first 500 iterations. (c) Zoom in and decaying sinusoidal fit for the 500^{th} to 3000^{th} iteration. (d) A different Fourier value from that shown in (a), (b) and (c).



(a) (b)

Figure 2.18 Value versus iterations for an (a) unconstrained and (b) constrained pixel in a 40×40 pixel image using the DM algorithm with $\beta = 0.7$. Note the difference in scale on the axes.

support, so no image space information exists on that pixel and it therefore takes much longer to converge. The pixel in Fig. 2.18(b) is outside the support, and so it is known that its value is 0, making the convergence much more rapid.

2.3.3 Non-convex constraints in higher dimensions

Non-convexity comes in many degrees of severity. Some non-convex constraints are locally convex, in the sense that on a small scale the constraint looks convex. For example, the Fourier magnitude constraint is locally convex. The plots of two Fourier values in a non-convex problem consisting of a support and a Fourier magnitude constraint are shown in Fig. 2.19. A damped sinusoid form persists, but the geometric decay ratio and the sinusoid frequency are no longer linked. In the two pixels used as examples, one has an effective angle of $\theta_{env} = 1.53°$ for the damping part but an effective angle of $\theta_{sin} = 0.77°$ for the sinusoidal part, while the other has effective angles of $\theta_{env} = 0.56°$ and $\theta_{sin} = 0.38°$. These effective angles are far enough apart while providing a very good fit of the curve, making it unlikely to be chance.

2.3.4 Increasing the speed of convergence

The simplest technique to increase the speed of convergence is to average the iterates over one damped sinusoidal cycle. However, if the cycle is long the averaging could take many iterations. Furthermore, the period of the cycle needs to be determined, and the periods for each of the pixels is different.

Another technique is to intersperse the DM recursions with some ER recursions, which if done at the correct point in the cycle can greatly speed convergence. However, it is very difficult to decide when to use the ER algorithm, and in many cases the best time at

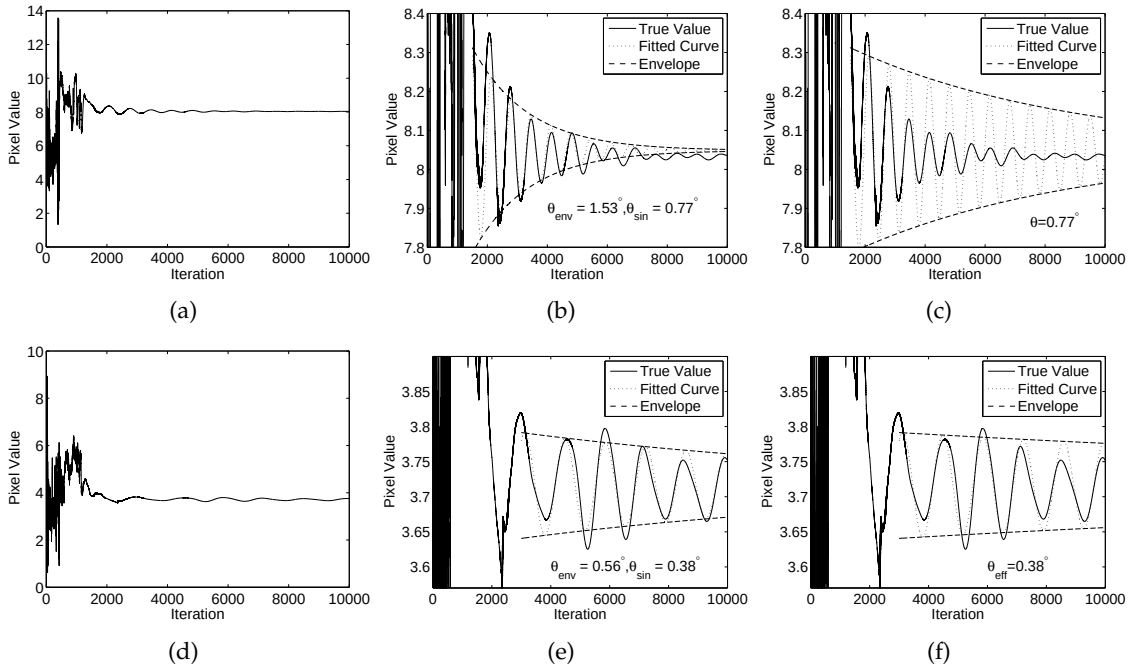


Figure 2.19 Line plots for a pixel in a non-convex problem. (a) All iterations. (b) Envelope and sinusoid fitted separately. (c) Envelope and sinusoid fitted together. Line plots for a second pixel. (d) All iterations. (e) Envelope and sinusoid fitted separately. (f) Envelope and sinusoid fitted together.

which to apply the ER algorithm is different for each pixel. The ER algorithms are therefore usually used every fixed number of DM iterations. Interestingly, mixing algorithms is a common technique in optics, where HIO recursions are mixed with ER recursions, and are shown empirically to increase the convergence speed.

In light of the studies above showing the problems caused by small "effective angles", a more sophisticated algorithm has been developed by the author, which is referred to as the Closest Point (CP) algorithm. This algorithm is designed to be immune to small "effective angle" problems and rapidly converge when such a situation arises.

2.3.4.1 Closest Point algorithm

Given any 4 points with two on each of two constraints, the four points select out a 3-D subspace of $\mathbb{R}^{2N}$, and it is possible to find the point at which the two lines formed by linearizing each pair of points most closely pass each other. This is illustrated in Fig. 2.20.

The two closest points are denoted E and F where

$$E = A + m(B - A), \quad m \in \mathbb{R} \tag{2.17}$$

$$F = C + n(D - C), \quad n \in \mathbb{R}. \tag{2.18}$$

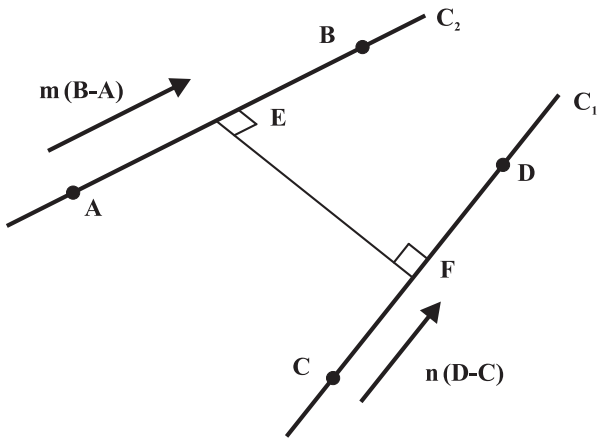


Figure 2.20 Closest Point Algorithm.

Noting that AE is orthogonal to EF and CF is orthogonal to EF, it is easy to show that

$$n = \frac{(B-A)\cdot(A-C)+m|B-A|^2}{(B-A)\cdot(D-C)} \tag{2.19}$$

$$m = \frac{(D-C)\cdot(A-C)-\frac{(B-A)\cdot(A-C)}{(B-A)\cdot(D-C)}|D-C|^2}{(C-D)\cdot(B-A)+\frac{|B-A|^2}{(B-A)\cdot(D-C)}|D-C|^2}. \tag{2.20}$$

Various methods can be used to choose the points A, B, C and D. For example, the four projections $P_A\mathbf{x}_n$, $P_B\mathbf{x}_n$, $P_AT_B\mathbf{x}_n$ and $P_BT_A\mathbf{x}_n$ used in the DM algorithm ($|\beta| \neq 1$) provide natural choices for the four points. The CP algorithm iteration is then given by

$$\mathbf{x}_{n+1} = \frac{E+F}{2}, \tag{2.21}$$

where E and F are as defined above, with $A = P_A\mathbf{x}_n$, $B = P_AT_B\mathbf{x}_n$, $C = P_B\mathbf{x}_n$, $D = P_BT_A\mathbf{x}_n$. Alternatively, only E or F may be used to save computational time. Note that if E or F are used then the new iterate already satisfies one of the constraints, so the projection from that iterate onto the relevant constraint need not be calculated, providing further computational savings.

The strength of the CP algorithm iteration is that it is immune to any small "angle" convergence problems that affect both the ER and DM IPAs and their variants, as will be demonstrated in Sec. 2.3.4.2. The weakness of the CP iteration is that it is very stable and stagnates easily. Successful search algorithms like the DM and its variants are unstable at near-solutions, which allows them to search the space. Therefore, the CP algorithm is best used once in a while as part of the normal DM iterates, and only after the algorithm has entered the convergence region and the error metric has dropped. While the CP recursion

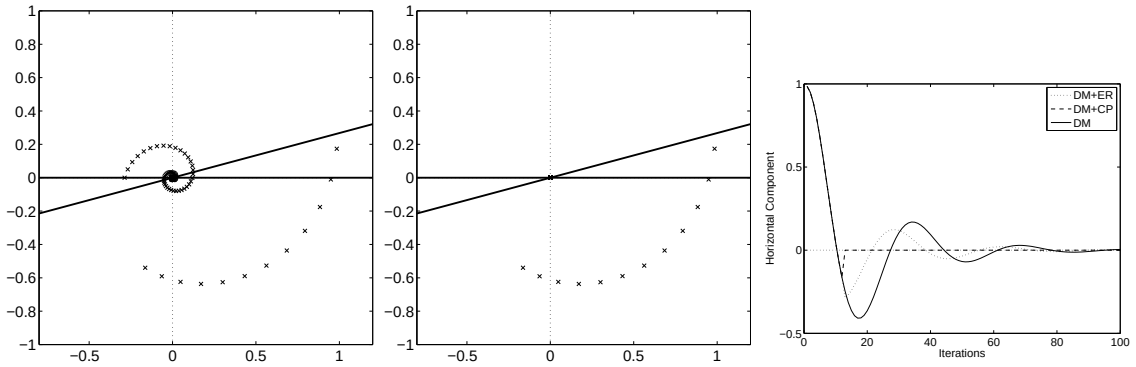


Figure 2.21 2D plots showing the results of DM iterations with one iteration of (a) ER and (b) CP at the 12^{th} iteration. (c) Line plot showing the x-axis value for the DM, DM+ER, and DM+CP algorithms.

is computationally expensive, the CP iterations converge or stagnate very quickly, so only around 3-5 consecutive iterations of the CP algorithm are needed each time.

The effect of the CP algorithm can be illustrated by example on the 2D problem from Sec. 2.3.1.1. An ER and CP recursion is now added to the DM algorithm at the 12^{th} iteration as shown in Fig. 2.21. The ER recursion helps to reduce the amplitude of the oscillation, but the effect is dependent on the point in the damped sinusoidal cycle where the ER recursion is used. The CP recursion shows its immunity to small angle convergence problems by converging instantly in one recursion.

When there is noise in the image, the multidimensional nature of the Euclidean space means that the two lines would be likely to become more orthogonal rather than parallel. So the CP iteration will make smaller steps, and so it tends towards being conservative and stable in the presence of noise.

2.3.4.2 Convergence of the CP algorithm

The CP algorithm is now applied to two example problems, one with all convex constraints and one with a non-convex constraint. The constraints for the convex problem are support and Fourier value constraints, and the constraints for the non-convex problem are support and Fourier magnitude constraints. Note that zero-dimensional constraints usually converge quickly with no small angle problems and therefore do not need to be considered. A 40×40 pixel image is used, and 5 iterations of the ER or CP recursion are applied every 100 iterations of the DM recursion. The RMS error versus iteration plots are shown in Fig. 2.22. Both hybrid algorithms performed better than only using the DM algorithm in the convergence phase of the algorithm, but the CP algorithm outperformed the ER algorithm. It was found that performance was best when the CP and ER steps are only added after approximate convergence, i.e. convergence to a point near the solution where the local geometry

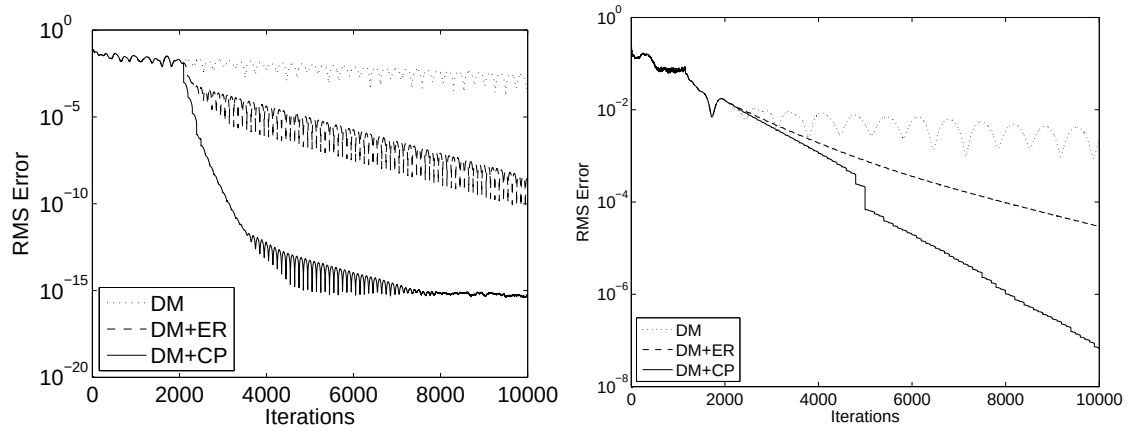


Figure 2.22 Local convergence speedup using the CP algorithm for (a) convex and (b) non-convex problems.

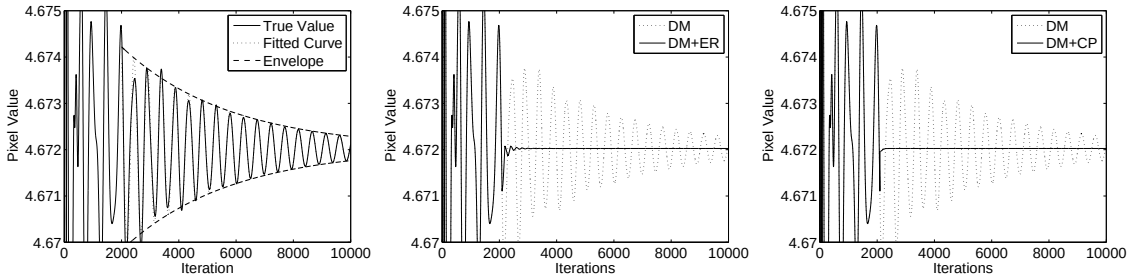


Figure 2.23 A single Fourier magnitude value versus iteration for the convex problem. (a) DM, (b) DM + ER, (c) DM+CP.

is approximately convex.

The value of a single pixel versus iteration for the convex problem is shown in Fig. 2.23. The plot shows the usual damped sinusoid. Interspersing some iterations of ER speeds up convergence as can be seen from the step change at the 2000^{th} iteration. However, the step was not precisely to the correct solution, and as a result more fluctuations in the form of a second damped sinusoid occur. With the CP algorithm the step is much more precise, and less fluctuations occur afterwards. The non-convex case is shown in Fig. 2.24. Despite no damped sinusoid form occurring, interspersing the ER and CP iterations also helps speed up convergence. Note that the damping ratio and sinusoidal frequency are different for the estimate, but they are the same for the iterate as shown in Fig. 2.24(a).

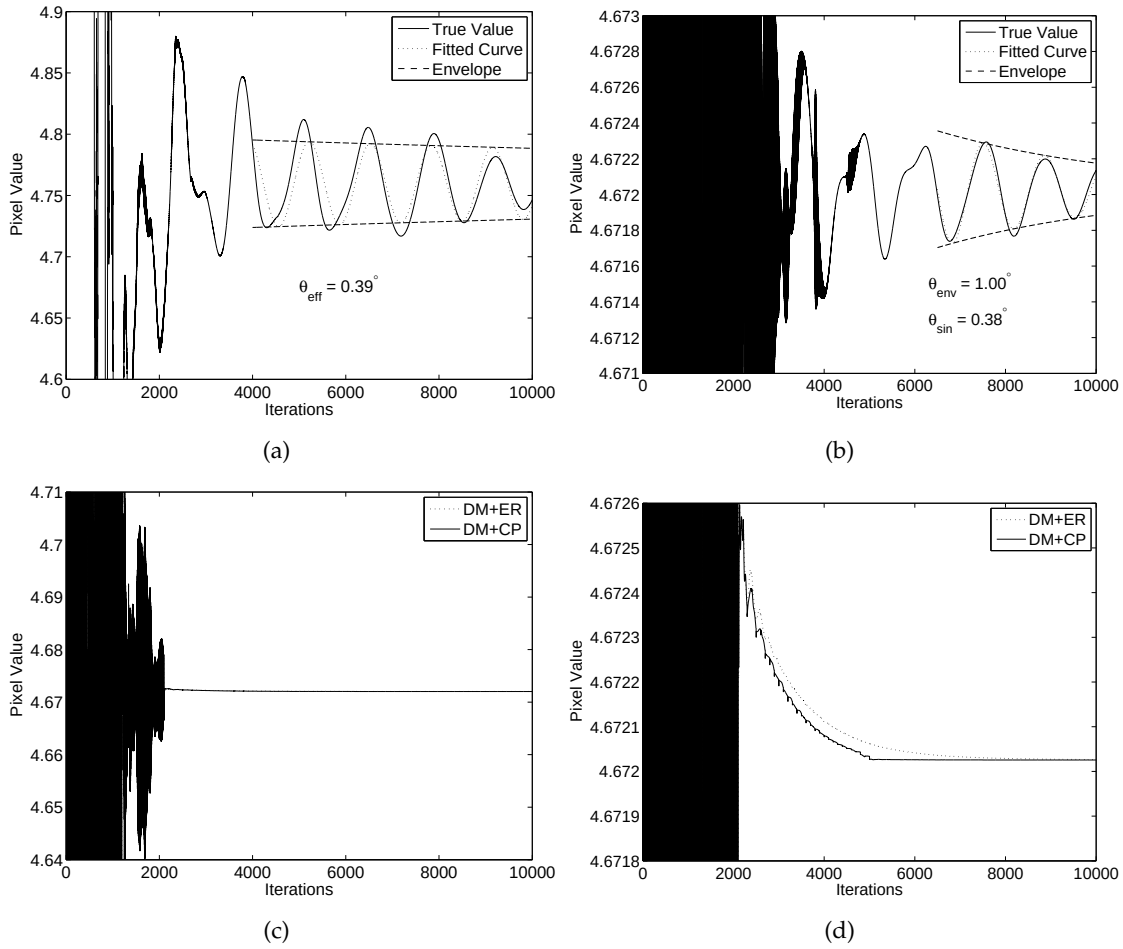


Figure 2.24 A single Fourier magnitude value versus iteration for the non-convex problem. (a) DM iterate, (b) DM estimate, (c) DM+ER and DM+CP estimate, which converges faster than DM alone. (d) A close up of (c) showing that the DM+CP algorithm converges faster than the DM+ER algorithm.

2.4 Desirable properties of an IPA

Most good IPAs move randomly around the search space, converging quickly when there is a solution or near solution, and moving away again if no fixed point is found. Some desirable properties for a good IPA are:

Rapid convergence Converges rapidly near a solution, reducing the number of iterations required. If the "effective angle" is small, the algorithm may take a long time to converge. In some problems, especially where the constraints are convex or "less non-convex", the region of convergence is quite large, and the algorithm is in the convergence phase for most of the iterations.

Large region of convergence A large region of convergence, which increases the probability of convergence. In the ideal case the region of convergence is the same as the search space, such as when both constraints are convex.

Iterate stays near the solution The iterate spends time near a solution, with the amount of time spent being related to the distance between the constraints at the solution. In practical problems, the presence of noise means that there is no intersection between the two (overconstrained) constraints. To have any hope of finding the solution, the distance between constraints at the true solution must be significantly smaller than the distance between the constraints in the vicinity of the false solutions. Then ideally the algorithm will explore the search space, spending some time at each candidate-solution before moving on to the next. In order to maximize the quality of the solution, the iterate should spend some time searching at the vicinity of a candidate-solution, with more time being spent at candidate-solutions with smaller distances between the constraints. Setting the length of time spent at each candidate-solution to be inversely proportional to the distance between constraints at the solution is a reasonable choice.

If the true solution has a much smaller distance between the constraints than the false solutions, the iterate will spend more time at the true solution, or may even be trapped there, so the answer can easily be determined. If the distance between constraints at the near solutions are of a similar size, then the user or a second algorithm will have to inspect all the candidate solutions to find the correct one.

In the DM algorithm it can be seen that the distance the iterate moves at each iteration is proportional to the distance between constraints and to β [16]. Thus the iterate moves a shorter distance when the distances between constraints is small, and therefore spends more time in the vicinity of near-solutions, with the length of time spent being inversely proportional to the distance between the constraints. The value of β should be chosen to be of the same order of magnitude as 1 as it is the "natural" scale of the distance between

the constraints. A smaller β will yield an algorithm that moves less at each iteration, so it is slower at searching but more stable. This is shown experimentally in Sec. 2.10.

Iterate does not get trapped at false solutions The iterate does not get trapped at a false solution, i.e. the movement of the iterate is not or does not tend to zero when it is not at or near the solution. If a projection algorithm is searching in a limited space, say of limited energy, and does not stagnate or get trapped in any limit cycle, then if a sufficient number of iterations is run, then assuming that there is a finite convergence region, the algorithm is guaranteed to find the correct solution eventually. While it is very difficult to find an algorithm which will not get stuck in a k-iteration limit cycle ($\mathbf{x}_{n+k} \approx \mathbf{x}_n$), an attempt can be made to limit the possibility of a 1-iteration limit cycle ($\mathbf{x}_{n+1} \approx \mathbf{x}_n$). For example, this is achieved in the difference map algorithm given by $\mathbf{x}_{n+1} = \mathbf{x}_n + P_A f_1 - P_B f_2$ where f_1 and f_2 are any functions. Then $||\mathbf{x}_{n+1} - \mathbf{x}_n|| = ||P_A f_1 - P_B f_2||$, so the iterate stagnates only if a solution has been found, and moves in small steps if and only if the iterate is near a solution, and when $\mathbf{x}_{n+1} = \mathbf{x}_n$ a solution has been found.

This behaviour can also be found in the relaxed projection (RP) algorithm as defined in Eq. 1.61. If $\mathbf{x}_{n+1} = \mathbf{x}_n$ and $\gamma_A \gamma_B = 1$, then at stagnation,

$$\begin{aligned} \mathbf{x}_n &= \mathbf{x}_{n+1} \\ &= T_B T_A \mathbf{x}_n \\ &= P'_B + \gamma_B (P'_B - (P_A + \gamma_A (P_A - \mathbf{x}))) \\ &= P'_B + \gamma_B P'_B - \gamma_B P_A - \gamma_B \gamma_A P_A + \gamma_B \gamma_A \mathbf{x}_n, \end{aligned} \tag{2.22}$$

so that

$$(1 + \gamma_B) P'_B = (1 + \gamma_B) P_A \tag{2.23}$$

$$P'_B = P_A \tag{2.24}$$

where $P'_B = P_B T_A \mathbf{x} = P_B (P_A + \gamma_A (P_A - \mathbf{x}))$. $P'_B = P_A$ is then the result of both projections and is therefore the solution. So the RP algorithm with $\gamma_A \gamma_B = 1$ only stagnates if the solution has been found, and is thus a reasonably good algorithm for searching the attractor. This is evidenced by its good results when used to solve the binary and Fourier magnitude problem, where the discrete binary constraint causes other algorithms to stagnate. However, it performs extremely poorly in the convergence phase with this choice of parameters, since the algorithm is "over-relaxed", so it takes too large a step at each iteration. For example, the RP algorithm with this choice of parameters fails to converge in the simple case of two linear constraint sets in 2D. This is less of an issue for the binary and Fourier magnitude problem due to the discrete binary constraint. Furthermore, it is prone to limit cycles of 2 or more iterations, but this problem can often be solved by averaging the iterate over the limit cycle. Interestingly, the DR algorithm is equivalent to the RP algorithm averaged over two cycles, which has the additional effect of fixing the convergence problem caused by the RP algorithm with $\gamma_A = \gamma_B = 1$ being over-relaxed.

Limited search space or large solution space This can be achieved in two ways.

a) By limiting the iterate to a subset of the Euclidean space which contains the true solution. For example, this can be done by keeping the iterate always inside one of the constraint sets. If the constraint A is a convex constraint of infinite extent with the all-zero image $\mathbf{0} \in A$, and the initial iterate $\mathbf{x}_0$ is in A, then the iterates of the recursion $\mathbf{x}_{n+1} = \mathbf{x}_n + \gamma_1 P_A f_1 \mathbf{x}_n + \gamma_2 P_A f_2 \mathbf{x}_n$ for any choice of functions f_1 and f_2 and parameters γ_1 and γ_2 will also remain in the constraint set A. An example of this is in crystallography, where the crystallographic symmetry constraint is a convex constraint set of infinite extent with $\mathbf{0} \in A$, and can therefore be used to limit the search space (note the fixed point dimensionality also decreases) and also to reduce the amount of computer memory needed to store the iterates and reduce the need to enforce A explicitly in the constraints.

b) Alteratively, one can search for a manifold instead of the 0-dimensional true solution. For example, the DM algorithm has a manifold of fixed points, and finding any of the fixed points is equivalent to finding the solution. In the DM algorithm the iterate is recalled to be given by

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \beta(P_A T_B \mathbf{x} - P_B T_A \mathbf{x}). \tag{2.25}$$

The iterate is at a fixed point when $\mathbf{x}_{n+1} = \mathbf{x}_n$. Then $P_A T_B \mathbf{x} = P_B T_A \mathbf{x}$ satisfies both constraints and is therefore the solution.

If there is an N-dimensional space and an N_A-dimensional constraint A, then there is an $(N - N_A)$-dimensional subspace which projects onto each point in A. Thus assuming the solution is a single point, there is an $(N - N_A)$-dimensional subspace which results in the solution when the $P_A T_B$ operation is applied to it. The intersection of the two subspaces, one for each constraint, is the dimensionality N_F of the fixed point set, and is given by

$$\begin{aligned} N_F &= (N - N_A) + (N - N_B) - N \\ &= N - (N_A + N_B). \end{aligned} \tag{2.26}$$

Thus in some sense the "solution space" is of dimension $N_A + N_B$.

2.5 Uniqueness and false solutions

The constraints sometimes do not uniquely define the solution, so the error metrics may be small, but a useful solution is not obtained. Some of these *false solutions* may be *trivially-related* to the true solution. For example, neither the Fourier magnitude nor binary constraint is able to distinguish a circularly shifted false solution from the true solution. These types of trivial false solutions are not generally a problem, and in any case it is impossible to distinguish them from the true solution using only the constraints.

A more serious problem is that of false solutions which bear no relation to the true solution. For an ideal algorithm, all possible information is included in the constraints, making it im-

possible to distinguish a false solution from a true solution. A common technique used in crystallography is therefore to withhold from the algorithm a few pieces of information, for example, a few Fourier magnitudes. The algorithm is then run, and the withheld Fourier magnitudes are then compared to the reconstructed ones to ensure the veracity of the solution. Such error metrics are referred to as the *free* error metrics.

Define a near-false solution as a solution which satisfies nearly all the constraints. A near-false solution is attractive to the algorithm, and can create traps. An example is if a binary-fill fraction constraint is used, along with a Fourier magnitude constraint. The inverse of the image, i.e. with the two values flipped, satisfies all values of the Fourier magnitude constraint and the binary constraint, but does not satisfy the fill fraction constraint unless $f = 0.5$. This can create an attractive point to the algorithm if $f \approx 0.5$. A solution to this problem is to invert the iterate and run the algorithm for some iterations once in a while. If the iterates are trapped due to an inverse solution, the algorithm will quickly converge to the true solution. This is further described in Sec. 3.3.3.

If an iterate is exactly in the middle between two equally attractive solutions, one true and one false, then the algorithm may be equally attracted to both, and convergence may be slow, or the algorithm may even be unable to converge to either of the solutions [27]. This is more likely to happen if the two solutions are duals of each other, such as an image and its inverted in the origin version. Rounding error can help break the deadlock, but another solution would be simply to start the iterations at an image biased heavily towards one or another of the solutions. This can be done by starting from a random image and setting one of the pixels to a very large value. The large pixel value is likely to break the symmetry between the image and its duals.

2.6 More than two constraints

Ideally, all information about the problem is input into the algorithm in the form of constraints. In most real world problems, this results in more than two constraints, which is a problem since most projection algorithms accept only two constraints.

2.6.1 Combining constraints to form one constraint

One way to reduce the number of constraints to two is to combine some of the constraints. The projection must then make the minimal change to the image such that it satisfies all of the combined constraints. In phase retrieval problems, for example, one set of constraints is in the image domain, and the other set is in the Fourier domain, giving a natural division of the constraints.

In the simplest case the constraints to be combined commute. If two constraints A and B commute, then $P_A P_B \mathbf{x} = P_B P_A \mathbf{x}$ $\forall \mathbf{x}$ and so the result satisfies both constraints. The resulting projection may be a projection onto $A \cap B$, i.e. $P_A P_B \mathbf{x} = P_B P_A \mathbf{x} = P_{A \cap B} \mathbf{x}$, or

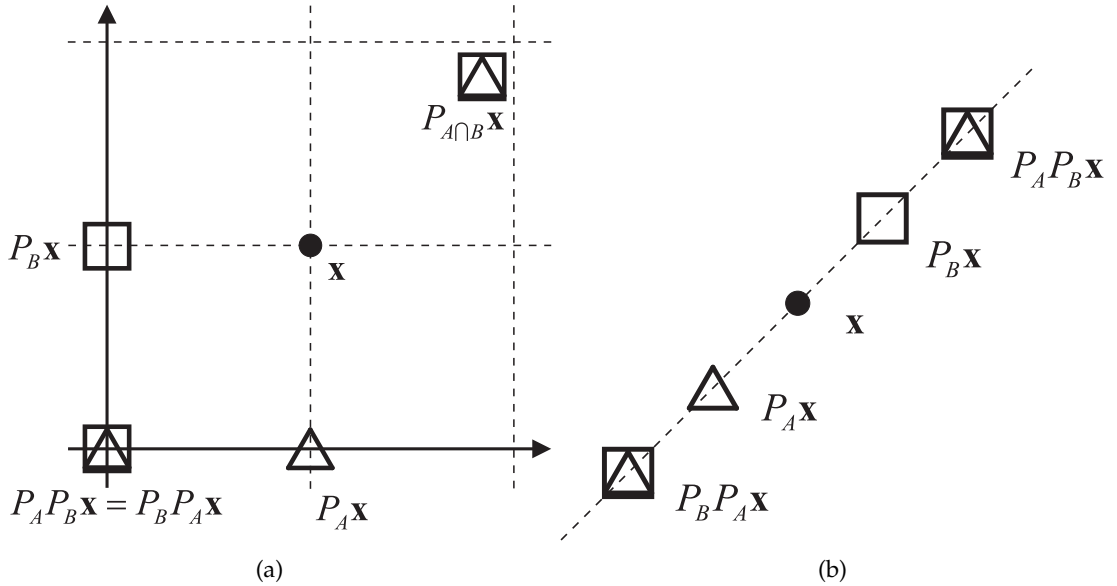


Figure 2.25 Some counterexamples. The △s represent one zero-dimensional constraint (A), and the □s represent the other (B). (a) The two constraints commute but the result of $P_AP_B\mathbf{x} = P_BP_A\mathbf{x}$ from the • is not the projection onto $A \cap B$. (b) The two constraints maintain each other, but $P_AP_B\mathbf{x} \neq P_BP_A\mathbf{x}$ from the •.

it may not be a projection, i.e. $P_AP_B\mathbf{x} = P_BP_A\mathbf{x} \neq P_{A\cap B}\mathbf{x}$. An example in 2-D where a pair of commuting constraints do not yield the projection when concatenated is shown in Fig. 2.25(a).

If two constraints commute, then each constraint is maintained when the other is applied to it, i.e. if $\mathbf{x}$ is in A, then $P_A\mathbf{x} = \mathbf{x}$, so

$$P_BP_A\mathbf{x} = P_B\mathbf{x} = P_AP_B\mathbf{x}, \tag{2.27}$$

so $P_B\mathbf{x}$ is in A. The same applies with $\mathbf{x}$ in B. Note that the converse is not true, so if $\mathbf{x}$ is in A implies $P_B\mathbf{x}$ is also in A, and $\mathbf{x}$ is in B implies $P_A\mathbf{x}$ is also in B, the two constraints may not commute. An example in 2-D of a pair of constraints which maintain each other but do not commute is shown in Fig. 2.25(b).

In some cases the constraints nearly-commute. For example, in Chapter 6 the support and interpolated symmetry projection commute with the possible exception of the edges. In this case the approximate projection formed by concatenation of the two projections also yields acceptable results.

If the constraints do not commute, one can use a second iterative projection algorithm to solve for the intersection of the two constraints. Then there is generally no guarantee that this is a projection, but if it is close enough the algorithm will often work regardless.

2.6.2 Constraints enforced directly upon the iterate

If a constraint C is a convex constraint of infinite extent and is maintained when projections onto all the constraints used in an algorithm are applied to it, and the projection algorithm itself is a linear combination of the individual projections, then the constraint C can be enforced directly upon the iterate, so that all iterates and projections satisfy the constraint C. This lowers the dimensionality of the search space by the dimensionality of C, $\dim(C)$, so although the dimensionality of the fixed point set is also lowered by $\dim(C)$, which may or may not help in convergence, efficiencies can be gained in the representation of the image and in the lack of need for enforcing the constraint C, which helps to speed up convergence. For example, the support, binary, connectivity, Fourier, and Fourier magnitude projections are all orthogonal, to and therefore maintain, crystallographic (global non-interpolated) symmetry. The binary fill fraction constraint maintains crystallographic symmetry if fN is a multiple of the order of the symmetry. Since all commonly used projection algorithms can be written as linear combinations of the projections, the crystallographic symmetry constraint can be directly enforced upon the iterate. Due to finite precision errors, it is prudent to continue to enforce the crystallographic constraint C on the iterate at the end of each iteration.

2.6.3 Critical constraints

Some constraints, or parameters, need to be estimated during the algorithm, but the parameters have too great an effect on the other parameters (pixel values). These are referred to here as *critical* constraints. An example is the scale factor s of the Fourier magnitudes in the binary and Fourier magnitude problem as discussed in Ch. 4. Critical parameters, if treated like normal pixel values and modified in accordance with the rules of the IPA recursion, will not lead to a successful searching and converging algorithm since the initial choice of the critical parameter influences the subsequent behaviour of the algorithm too much. The solution is to estimate these parameters, and only change them after the algorithm has had a chance to search the space and attempted to converge. The re-estimate can be carried out using the current best estimate of the solution or by attempting to search all possible values of the critical parameter.

2.6.4 Divide and Concur

If the constraints are not easily combined, the "divide and concur" method [28] allows IPAs to accept any number of constraints at the cost of increased memory usage. In this method, q copies of the image are made, one for each of the q constraints. The application of the q projections to each of the copies creates the first projection set. The second constraint is the constraint that all q copies of the image are identical, so the projection is to average all q copies. Note that if all q constraints are convex, then the combined constraint is also convex. Furthermore, the averaging constraint is a convex constraint of infinite extent.

2.6.5 Algorithms which accept more than one constraint

For some simple problems such as the all-convex case, the generalized ER algorithm given by $\mathbf{x}_{n+1} = P_q ... P_2 P_1 \mathbf{x}_n$ is able to find a solution. It is also simple to generalize the relaxed projection algorithm in this way.

2.7 Resolution extension (bootstrapping)

Resolution extension, or *bootstrapping*, is a common procedure in crystallography (in which it is referred to as "phase extension") where the Fourier magnitudes are known. Initially, low resolution data is used to reconstruct the image, and then the solution is refined as more of the high resolution data is added in stages. Each cycle of converging to a solution and then adding higher resolution data is referred to here as a bootstrap. The higher resolution data can be zeroed out, low pass filtered, or a lower resolution grid can be used.

Bootstrapping is useful for less sophisticated algorithms such as the ER algorithm which have a small radius of convergence and stagnate often, i.e. with poor ability to search the space. The limited search space created by the removal of high resolution data means that the algorithm is able to find a reasonable initial estimate, with manual selection often used to choose a good initial estimate. The estimate is then improved by stepping out the high resolution limit, sometimes one resolution pixel at a time.

The use of bootstrapping with more sophisticated algorithms which have good search properties is less important, but can improve the reliability of finding the solution since it alleviates the problem of the algorithm being in the wrong attractor. It can also be useful if the lower resolution reconstructions have less noise (e.g. interpolation is being used), so that when the higher resolutions are phased, the iterate is already close to the true solution.

The *bootstrapping function* is the function which is multiplied element by element with the Fourier magnitudes to effect the low-pass filtering. The image which satisfies the constraints at each bootstrap is the original image convolved with the inverse Fourier transform of the bootstrapping function, so the effect of the bootstrapping on the image domain constraints must be considered.

If the higher resolution data are zeroed out, this is equivalent to convolving the image with a sinc function. The sinc function is negative in some places, so the positivity constraint may no longer apply. Furthermore, a sinc function decays as the inverse of distance from the origin, so the support constraint will also be violated. A similar situation occurs when using a lower resolution grid.

A solution is to low-pass filter the image using a filter with a smoother roll-off and a non-negative impulse response. A Gaussian filter is a reasonable choice and is used later in this thesis. A popular error plot in crystallography is to plot error metric versus resolution

at each bootstrap. This requires a sharper cutoff at each frequency to be meaningful, so a filter such as a raised cosine filter may be used instead.

Each bootstrap is generally started with the estimate of the image from the previous bootstrap. Correcting for the windowing function is usually unstable due to the small values of the windowing function at the higher resolutions.

An added benefit of bootstrapping is that a reconstruction is made at each resolution. So if the higher resolution data are poor or if the sampling rate is too low for the interpolation to work well, a reconstruction will still be available, albeit at a lower resolution.

2.8 Estimating the solution

When the solution is found, the algorithm must be halted and a final estimate must be made. If there is no noise, the iterates will reach a fixed point. This can be easily detected and the algorithm halted. The image can then be perfectly reconstructed, usually by using one of the projections.

When there is noise in the data, the iterates will not reach a fixed point, but will continue to move around. The algorithm has converged in the sense than no significant further improvement of the estimate is possible, and if the noise is large enough, the iterates may even leave the vicinity of the solution.

An IPA therefore needs to detect when convergence is reached so that the algorithm can be halted and a final estimate of the solution made.

2.8.1 Detection of convergence

The simplest method for halting the algorithm is to simply halt after some large number of iterations. However, a lot of computer time will have been wasted if the solution has already been found, or the algorithm may have halted before a solution has been found, or it may have moved away from the solution.

When the algorithm enters the convergence region, there is a distinct fall in the error metrics, and metrics will often then stay low for further iterations, but can sometimes increase again if the noise levels are sufficiently high so that the solution is not attractive enough. In general it is difficult to predict the level to which the error will fall beforehand, and it is easier to detect a change in the error metrics. For example, the average error metric over the last $2k$ to k iterations can be compared with the average error metric over the last k iterations. This method is used in Ch. 6 for detecting convergence at each resolution extension, which is another application where detection of convergence is required.

2.8.2 The final estimate

2.8.2.1 Averaging

Once convergence has been detected, the final solution can be set to be the average of all or some subset of the estimates at each iteration after convergence. Features of the image which are consistent with the data will be in all of the images, and features which are "random" and essentially an artifact of the choice of initialization of the algorithm will be averaged out.

Using averaging produces very good results, but detection of when, or if, the algorithm has converged is necessary. Furthermore, detection of when the algorithm leaves the vicinity of the solution is also necessary.

2.8.2.2 Best metric

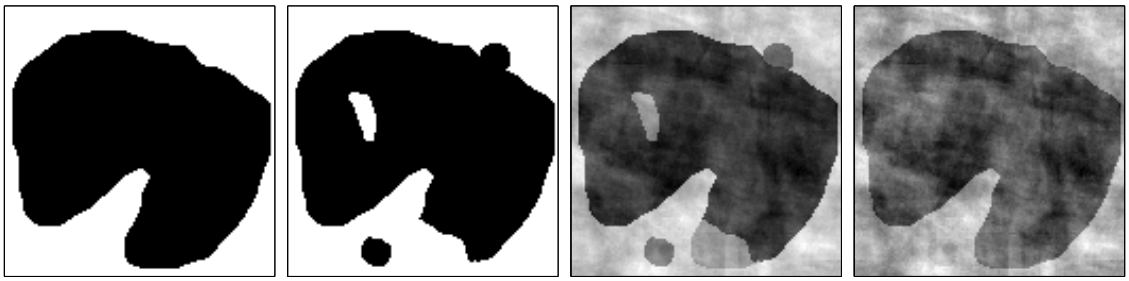
An alternative technique for estimating the final solution is to simply use the estimate of the iteration with the lowest error metric. The first advantage is that other than for halting the algorithm, detection of convergence is not needed. Secondly, the "best estimate so far" is available while the algorithm is running, so the algorithm can be halted if the solution is acceptable, and if the algorithm is halted prematurely there may still be a reasonable result. Thirdly, if the estimate is found from one of the projections as is often the case, the final estimate can be guaranteed to satisfy the constraint set for that projection. The disadvantage of this technique is that the final result may be slightly worse than when averaging is used.

2.8.2.3 Using a second algorithm for refinement

After convergence, a second algorithm may be used for refinement. The second algorithm is sometimes also a projection algorithm, but it may be a probabilistic algorithm or some other method of refinement.

An example is shown in Chapter 6, where interpolation is necessary due to one of the constraints. Using a very fine grid spacing gives better results but requires too much computational time, so the algorithm is run using a coarser grid spacing and resampled into a finer grid spacing after the algorithm has converged for the coarser case.

When algorithms like the HIO algorithm were first used, the difference between fixed points and solutions was not well understood, and the HIO algorithms were often ended with a few ER iterations to obtain a solution. This works reasonably well in practice, but is not as good as using one of the projections $P_A T_B \mathbf{x}$ or $P_B T_A \mathbf{x}$.



(a) (b) (c) (d)

Figure 2.26 "$2F_o - F_c$" map (a) Correct Image (b) $\mathbf{x}$ (c) $P_F\mathbf{x}$ (d) $2P_F\mathbf{x} - \mathbf{x}$.

2.8.3 Relaxed projections as an estimator

This thesis is primarily concerned with crystallography in which there is a Fourier magnitude constraint and some image domain constraints. A common technique in crystallography is to use the relaxed Fourier magnitude projection $2P_{FTMag}\mathbf{x} - \mathbf{x}$ as an estimate of the solution. This is often referred to as calculating the "$2F_o - F_c$" map [2] and is recognition of the fact that the relaxed Fourier magnitude projection is an excellent estimate of the solution if the iterate is close to the solution.

An example of using the "$2F_o - F_c$" map is shown in Fig. 2.26. The result of the relaxed Fourier magnitude projection is shown in Fig. 2.26(d) and can be seen to be an excellent estimate of the solution.

When the iterate has converged to a fixed point in the DM algorithm, the solution is given by $\hat{\mathbf{x}} = P_A T_B \mathbf{x} = P_B T_A \mathbf{x}$. If the constraints used in the algorithm are the Fourier magnitude constraint and an image domain constraint, then

$$\hat{\mathbf{x}} = P_I T_F \mathbf{x} = P_I[(1 + \gamma_F)P_F\mathbf{x} - (1/\gamma_F)\mathbf{x}]. \tag{2.28}$$

where P_I denotes the image domain projection and P_F denotes the Fourier magnitude projection. If $\gamma_F \approx 1$, then the step is equivalent to calculating the "$2F_o - F_c$" map and then applying the image space projection to further improve the estimate. This makes $P_I T_F \mathbf{x}$ an excellent estimator for the solution, and it is therefore a good idea to choose β and γ to take advantage of this effect in the absence of any evidence for another choice of parameters. Note that enforcing the image space constraint is desirable as the atomic structure is found using the image domain representation of the object, so it is important that the object satisfies the image domain constraints.

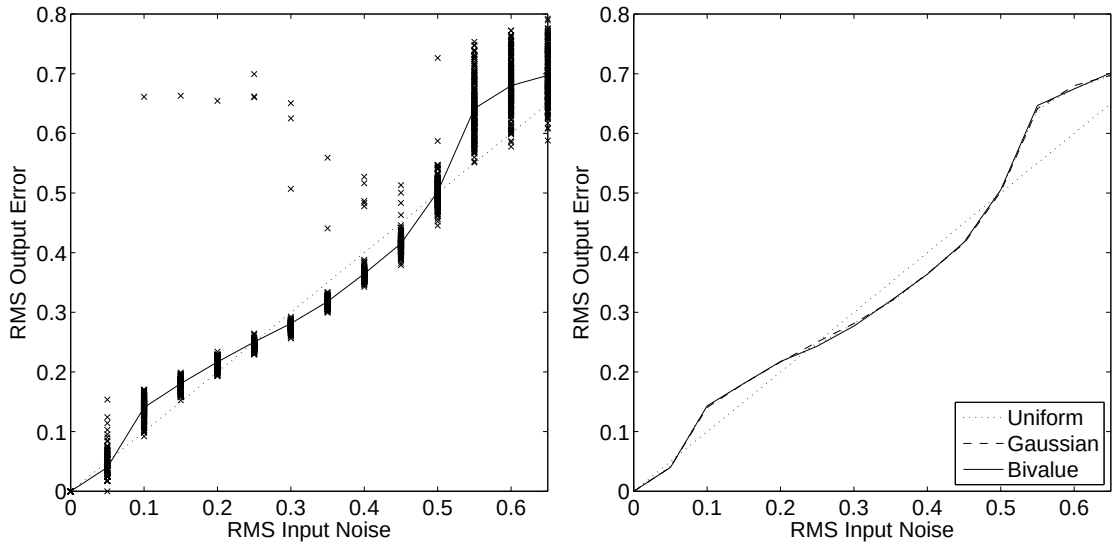


Figure 2.27 RMS input noise vs RMS output error. (a) Gaussian Noise. The outliers represent runs which failed to converge. (b) All three distributions plotted together.

2.9 Noise

In all real-world situations, noise is present in one or more of the constraints. In the Euclidean space, the addition of noise translates to a movement of the constraints. With the presence of noise the constraint sets may not all intersect, and so the algorithm will not reach a fixed point. However, there is still a marked drop in the error metric when the iterate reaches the vicinity of the solution. Since there is no fixed point, the algorithm is not guaranteed to stay near the solution, and may move away. The second effect of noise is that it will degrade the quality of the solution.

To illustrate the effect of noise, various amounts of uniform, Gaussian and bivalue noise were added to the image shown in Fig. 2.2, and the Fourier magnitudes are calculated from the noisy image. The Gaussian and uniform noise consists of adding Gaussian and uniformly distributed values respectively to each pixel in the image, and the bivalue noise consists of adding a constant positive or negative value to each pixel in the image. Using the binary fill fraction and Fourier magnitude constraints, 200 reconstructions were run at each noise level for 300 iterations, and the error in the reconstruction versus the noise level is shown in Fig. 2.27. The dotted straight lines in the two figures are the lines corresponding to equal input noise and output error energy. The distributions of the output reconstruction error along with their mean values are shown in Fig. 2.27(a) for the Gaussian case. The distributions are similar for the binary and uniform cases and are not shown. The mean reconstruction error versus input noise level for all three noise types is shown in Fig. 2.27(b), where they can be seen to be very similar. For these particular constraints, the error in the reconstruction is approximately the same as the noise in the input.

The error in the reconstruction is similar for all three noise distributions, with the fluctuations in the error plot likely caused by the initialization of the algorithm which was the same for all three distributions. This suggests that the RMS power of the noise is a key predictor of the performance of the algorithm in reconstruction, and the distribution of the noise is not as significant. Note that this is only true up to a certain noise level. For example, for a large level of binary noise the likelihood of an erroneous binary image matching the data better than the true image is much higher than for the other noise distributions. Furthermore, the quality of the reconstruction is quite consistent for each noise level, with the algorithm performing reasonably consistently for each of the 200 runs.

If many sources of noise are added together, then by the central limit theorem the overall noise will approach a Gaussian distribution. For this reason, when experimental data are not available, noise in this thesis is simulated by adding Gaussian noise to the image.

In the Euclidean space $\mathbb{R}^{2N}$, the pdf of Gaussian noise is a hypersphere with a Gaussian cross section. Since the image and Fourier domain representations are simply different basis functions of $\mathbb{R}^{2N}$, the Gaussian noise has identical effects in both domains. The Gaussian noise is equivalent to adding Rayleigh noise to the Fourier magnitudes. The direct addition of Gaussian noise to the Fourier magnitudes is not appropriate since it may cause the smaller Fourier magnitudes to be negative unless the noise level is low.

2.10 Stabilizing a projection algorithm

Since algorithms with good search abilities move away from near-solutions, the stability of the algorithm near the true solution can start to be a problem for high noise levels. If the algorithm is unstable at the solution, then it is unlikely that the algorithm will find and recognize the solution.

For very high noise cases with good noise characterization, Bayesian methods or a combination of projection algorithms and Bayesian methods, either sequentially or combined [29] may be more effective in finding the maximum likelihood solution.

A technique which has been found to be useful with projection algorithms is to linearly combine the iterate with a "stabilizing" image as an extra step at the end of each iteration, i.e.

$$\mathbf{x}_{n+1} = (1-\alpha)\mathbf{x}'_{n+1} + \alpha\mathbf{y}_n \tag{2.29}$$

where $\mathbf{x}'_{n+1}$ is the output of the usual projection algorithm recursion, $\mathbf{y}_n$ is the stabilizing image and α is a parameter set between 0 and 1. A variety of choices can be used for $\mathbf{y}_n$. For example, using the difference map and with $\mathbf{y}_n = P_B\mathbf{x}_n$, the overall update rule is given by

$$\mathbf{x}_{n+1} = (1-\alpha)[\mathbf{x}_n + \beta[P_A(P_B\mathbf{x} + \gamma_B(P_B\mathbf{x} - \mathbf{x})) - P_B(P_A\mathbf{x} + \gamma_A(P_A\mathbf{x} - \mathbf{x}))]] + (\alpha)P_B\mathbf{x}. \tag{2.30}$$

Comparing Eq. (2.30) with the RAAR equation given in Eq. (1.69), it can be seen that if $\beta = 1$, $\gamma_A = -1$, $\gamma_B = 1$ (i.e. the DR algorithm) with P_A as a support constraint and P_B as a Fourier magnitude constraint, then Eq. (2.30) is equivalent to the RAAR algorithm [19].

If $\mathbf{y}_n$ is set to $\mathbf{y}_n = P_B\mathbf{x}_n$ or a similar projection on $\mathbf{x}_n$, then setting $\alpha = 1$ results in the very simple algorithm $\mathbf{x}_{n+1} = \mathbf{y}_n$. This may be very stable but has no ability to explore the search space.

The situation is better if $\mathbf{y}_n = P_B\mathbf{x}'_{n+1}$, but in general the search capability is reduced by setting α large. Similarly, setting a small value of β will also increase stability but at the cost of searching power.

2.10.1 Experiments

The stabilization methods described above are explored by simulation using binary and Fourier magnitude constraints and a 128×128 pixel image. Large amounts of Gaussian noise were added to the image to produce noisy Fourier magnitude data. The estimated image was found by taking the projection $P_AT_B\mathbf{x}_n$. The quoted SNR of the noise is the ratio of the RMS noise power to the RMS image power. The algorithm was started at a point with an RMS error of 1×10^{-7} to the solution, which is essentially at the solution, and divergence of the iterate from the solution was observed. The DM algorithm was used as the main algorithm with various choices of $\mathbf{y}_n$ as shown in Table 2.1, where P_A denotes the binary projection and P_B the Fourier magnitude projection. It is worth noting that the stabilizers $\mathbf{y}_n = P_A\mathbf{x}'_{n+1}$ and $\mathbf{y}_n = P_B\mathbf{x}'_{n+1}$ require an additional projection operation. Various choices of $0 \leq \alpha \leq 1$ and $-1.2 \leq \beta \leq 1.2$ were used, and 10 runs of 100 iterations were run for each combination of parameters.

Table 2.1 Choices of stabilizing functions.

Stabilizer	$\mathbf{y}_n$
1	$P_A\mathbf{x}'_{n+1}$
2	$P_B\mathbf{x}'_{n+1}$
3	$P_A\mathbf{x}_n$
4	$P_B\mathbf{x}_n$
5	$P_AT_B\mathbf{x}_n$
6	$P_BT_A\mathbf{x}_n$

Setting $\alpha = 0.5$, $\beta = -0.5$, RMS Noise = 1, and $\mathbf{y}_n = P_A\mathbf{x}'_{n+1}$ (Case 1), the image error versus iterations are shown in Fig. 2.28. It can be seen that the runs are quite similar to each other, and this trend holds for different choices of parameters. The mean error across

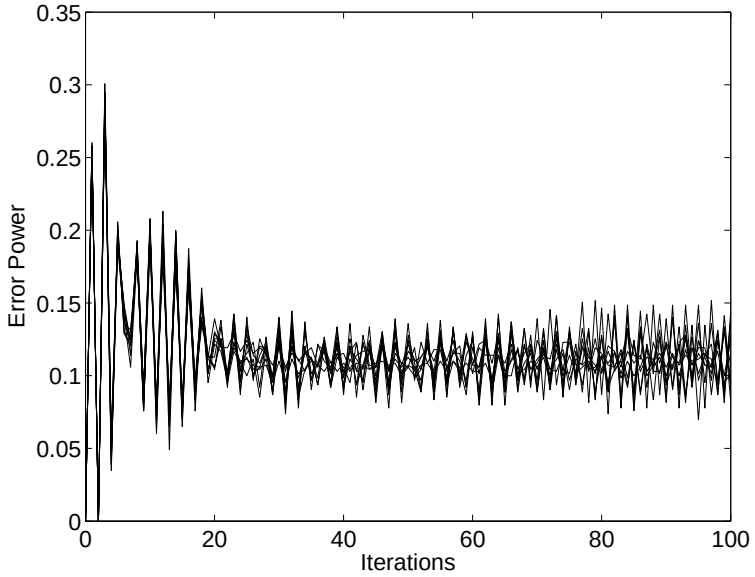


Figure 2.28 Image error vs Iterations for stabilizer 1 ($\mathbf{y}_n = P_A\mathbf{x}'_{n+1}$) for 10 runs with $\alpha = 0.5$, $\beta = -0.5$ and RMS Noise = 1.

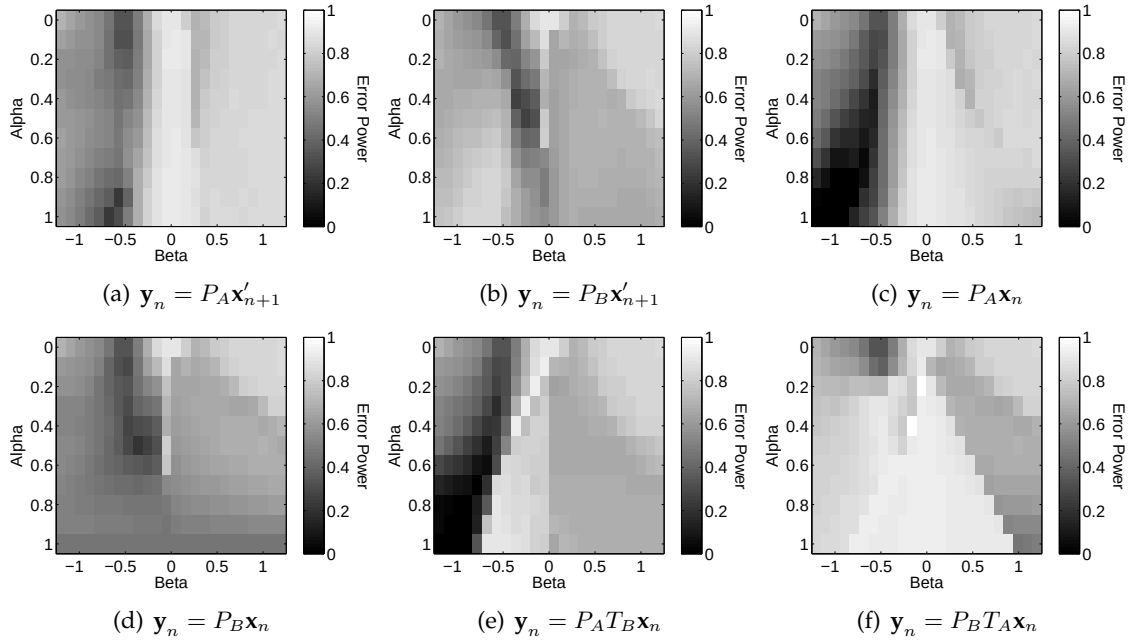
the runs at the 100^{th} iteration is therefore used as a metric of the stability of the algorithm for each set of parameters.

The effect of α and β on the stability is studied by varying the two parameters while using an SNR of 1 for each choice of $\mathbf{y}_n$. Fig. 2.29 shows plots of the mean image domain error after 100 iterations for each combination of α and β with a different choice of stabilizer for each subfigure. The greyscale represents the RMS error as shown in the colour bar.

Setting the SNR to 0.5, Fig. 2.30 shows the same plot as Fig. 2.29. The graphs are similar to that in Fig. 2.29, suggesting that moderate levels of SNR do not significantly affect the optimum choice of algorithm parameters.

In general, negative values of β performed better, with $\beta \approx -0.5$ performing the best. The very low error values with $\beta = -1$ and $\alpha = 1$ in Fig. 2.29(c) and Fig. 2.29(e) correspond to the algorithm $\mathbf{x}_{n+1} = P_A\mathbf{x}_n$ which has poor global convergence properties and is therefore not useful. Similarly, $\beta = 1$ and $\alpha = 1$ in Fig. 2.29(d) and Fig. 2.29(f) correspond to the algorithm $\mathbf{x}_{n+1} = P_B\mathbf{x}_n$. Furthermore, if $\beta = 1$, $\alpha = 1$, and $\mathbf{y}_n = P_B\mathbf{x}$ as in Fig. 2.29(b), the algorithm corresponds to the ER algorithm $\mathbf{x}_{n+1} = P_BP_A\mathbf{x}_n$ if P_B is idempotent. The same applies for $\beta = -1$, $\alpha = 1$, and $\mathbf{y}_n = P_A\mathbf{x}$ in Fig. 2.29(a). Note that since the noise is added to the Fourier magnitudes and the image error is being compared with the true binary image, the binary projection is noise-free while the Fourier magnitude projection is noisy.

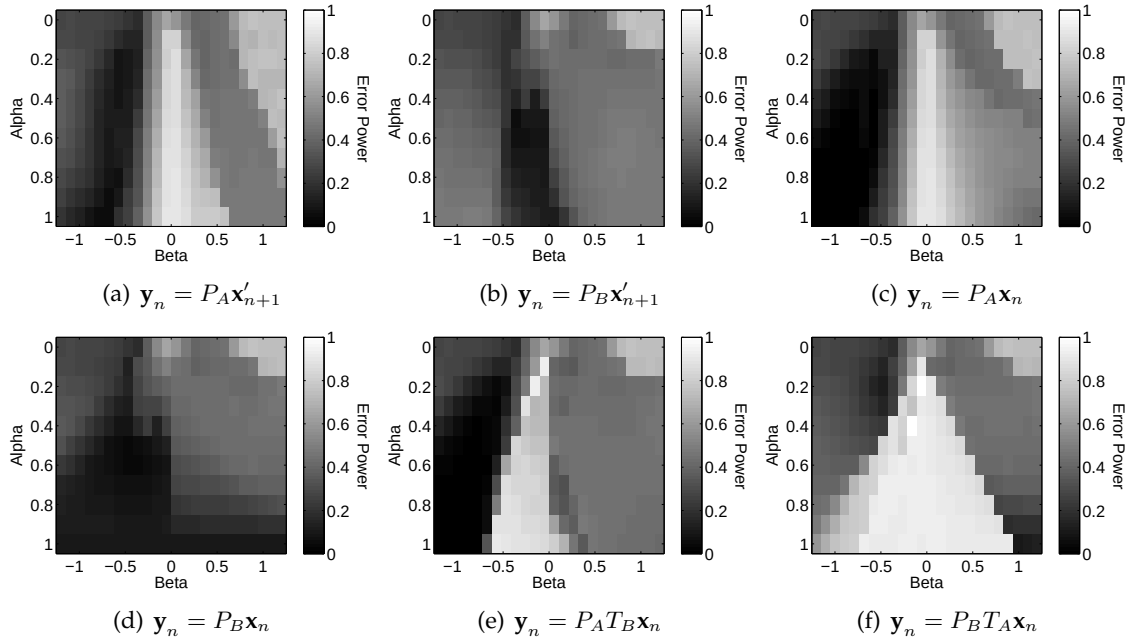
Setting the "optimum" value of $\beta = -0.5$, α and the SNR are varied, and the mean error at 100 iterations is shown in Fig. 2.31. Taking sections of Fig. 2.31, the mean error at 100



(a) $\mathbf{y}_n = P_A \mathbf{x}'_{n+1}$ (b) $\mathbf{y}_n = P_B \mathbf{x}'_{n+1}$ (c) $\mathbf{y}_n = P_A \mathbf{x}_n$

(d) $\mathbf{y}_n = P_B \mathbf{x}_n$ (e) $\mathbf{y}_n = P_A T_B \mathbf{x}_n$ (f) $\mathbf{y}_n = P_B T_A \mathbf{x}_n$

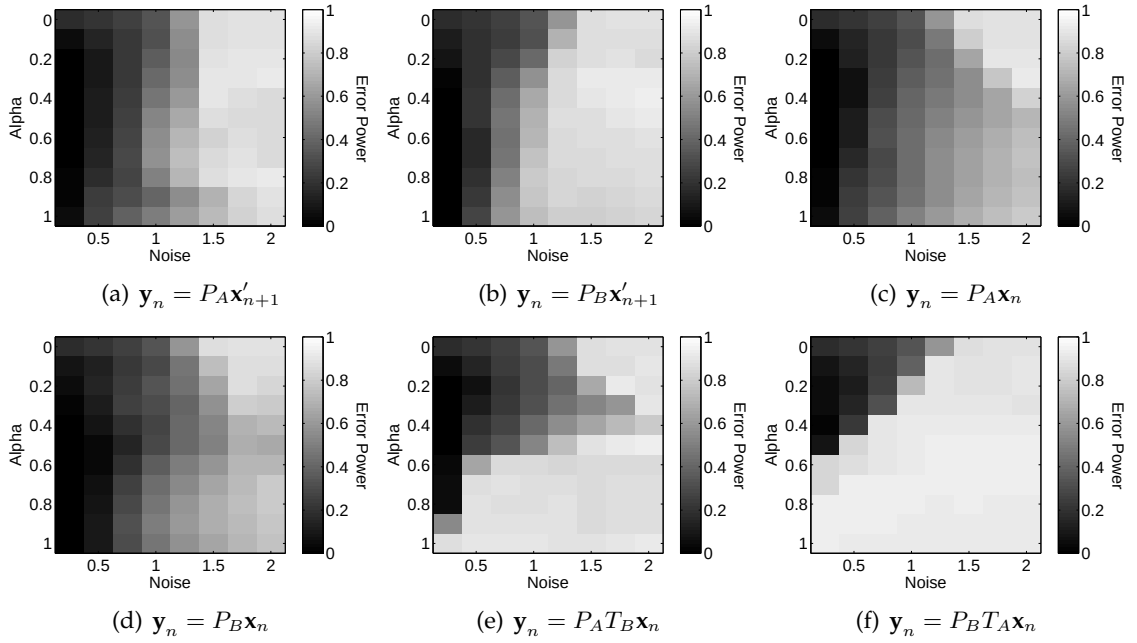
Figure 2.29 Mean image error at 100 iterations vs α and β with RMS Noise of 1 for different choices of the stabilizing function as shown.



(a) $\mathbf{y}_n = P_A \mathbf{x}'_{n+1}$ (b) $\mathbf{y}_n = P_B \mathbf{x}'_{n+1}$ (c) $\mathbf{y}_n = P_A \mathbf{x}_n$

(d) $\mathbf{y}_n = P_B \mathbf{x}_n$ (e) $\mathbf{y}_n = P_A T_B \mathbf{x}_n$ (f) $\mathbf{y}_n = P_B T_A \mathbf{x}_n$

Figure 2.30 Mean image error at 100 iterations vs α and β with RMS Noise of 0.5 for different choices of the stabilizing function as shown.



(a) $\mathbf{y}_n = P_A \mathbf{x}'_{n+1}$ (b) $\mathbf{y}_n = P_B \mathbf{x}'_{n+1}$ (c) $\mathbf{y}_n = P_A \mathbf{x}_n$

(d) $\mathbf{y}_n = P_B \mathbf{x}_n$ (e) $\mathbf{y}_n = P_A T_B \mathbf{x}_n$ (f) $\mathbf{y}_n = P_B T_A \mathbf{x}_n$

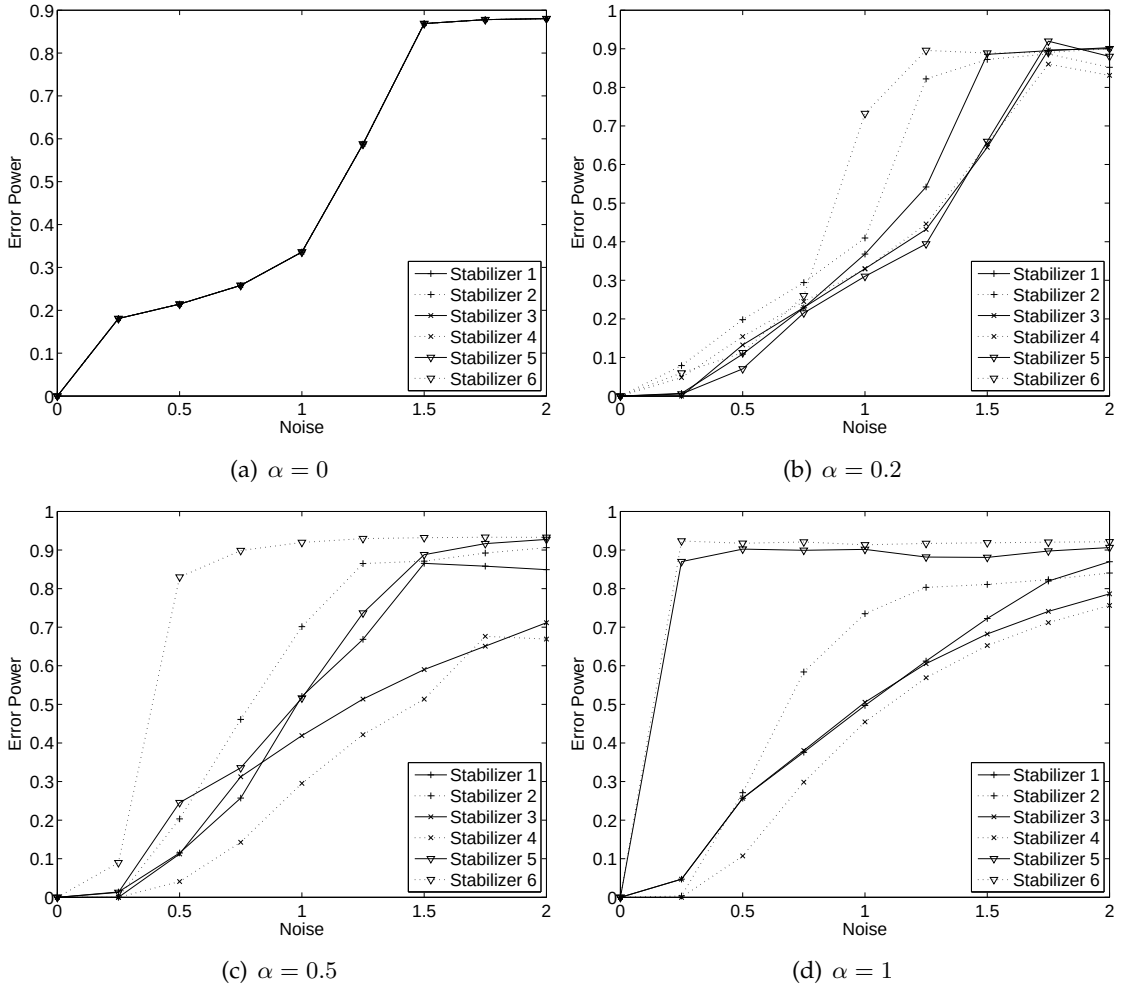
Figure 2.31 Mean image error at 100 iterations vs α and noise with $\beta = -0.5$ for different choices of the stabilizing function as shown.

iterations versus noise for a variety of stabilizing functions for different α with $\beta = -0.5$ are shown in Fig. 2.32. The plot for $\alpha = 0$ in Fig. 2.32(a) is identical for all choices of $\mathbf{y}_n$ since it is the standard DM algorithm. The choice of $\mathbf{y}_n = P_B \mathbf{x}_n$ performed the best, with $\mathbf{y}_n = P_A \mathbf{x}_n$ not far behind. The choice of $\mathbf{y}_n = P_B \mathbf{x}_n$ with an image-domain support constraint is identical to RAAR algorithm with a binary constraint instead of a support constraint.

Using optimized values of α and β for each stabilizing function, the mean error at 100 iterations versus noise for a variety of stabilizing functions is shown in Fig. 2.33. Values close to $\beta = -1$ and $\alpha = 1$ were disallowed for $\mathbf{y}_n = P_A \mathbf{x}_n$ and $\mathbf{y}_n = P_A T_B \mathbf{x}_n$ since these choices of parameters result in algorithms with poor global convergence properties as described previously. In general, using $\mathbf{y}_n = P_A \mathbf{x}_n$ and $\mathbf{y}_n = P_B \mathbf{x}_n$ worked well. Interestingly, these choices of stabilizing functions correspond to the RAAR algorithm. The stabilizer $\mathbf{y}_n = P_A \mathbf{x}'_{n+1}$ and $\mathbf{y}_n = P_B \mathbf{x}'_{n+1}$ also seemed to perform well. Using $\mathbf{y}_n = P_A T_B \mathbf{x}_n$ and $\mathbf{y}_n = P_B T_A \mathbf{x}_n$ were the worst options. The choice of parameters and stabilizing function which have proven to work well for this particular problem may not be optimal for another problem, but a similar convergence study could be done.

2.11 Projection algorithms as a map

The current position of the iterate completely determines the position of the next iterate. Therefore, each iteration of an algorithm is a map from $\mathbb{R}^{2N}$ to a set whose size is less than



(a) $\alpha = 0$ (b) $\alpha = 0.2$

(c) $\alpha = 0.5$ (d) $\alpha = 1$

Figure 2.32 Mean image error at 100 iterations vs noise with $\beta = -0.5$ for a variety of stabilizing functions with (a) $\alpha = 0$ (b) $\alpha = 0.2$ (c) $\alpha = 0.5$ (d) $\alpha = 1$. Stabilizers 1-6 are described in Table 2.1.

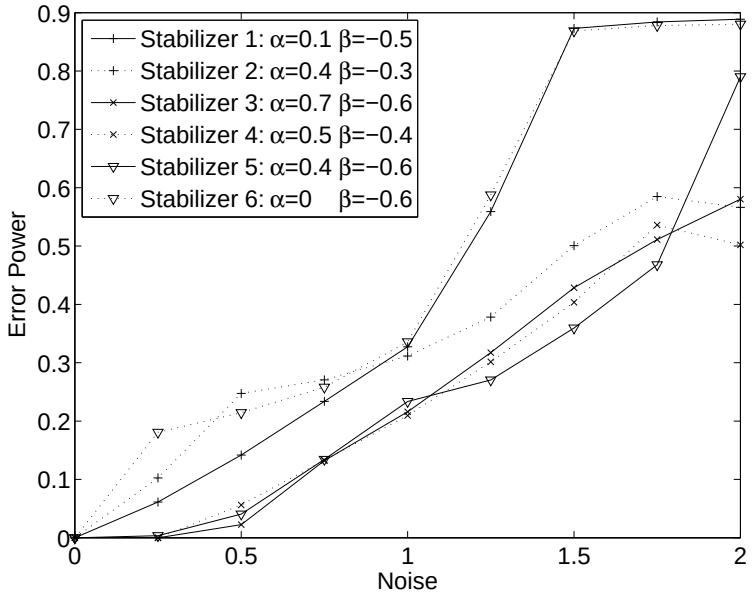


Figure 2.33 Mean image error at 100 iterations vs noise with α and β optimized for each choice of stabilizing function.

or equal to that of $\mathbb{R}^{2N}$, and the algorithm and constraint spaces are completely generalized by the map in the sense that the behaviour of the algorithm is governed solely by the map. The map can therefore be inverted to create a one-to-many function. So in principle by starting at the solutions or fixed points and repeatedly applying the inverted map operations, $\mathbb{R}^{2N}$ can be partitioned into sets which will or will not result in the solution, i.e. which are attracted to the correct or wrong attractors respectively. Note that this partitioning can also be carried out by starting from every point in $\mathbb{R}^{2N}$ and running the algorithm until convergence.

Each projection is a map from $\mathbb{R}^{2N}$ to C, where C is the constraint space. It is clear that any point on the line segment between an iterate $\mathbf{x}$ and its projection $P_C\mathbf{x}$ will also project to $P_C\mathbf{x}$, i.e.

$$P_C((1-\alpha)P_C\mathbf{x} - \alpha\mathbf{x}) = P_C\mathbf{x} \quad \text{for } 0 \leq \alpha \leq 1. \tag{2.31}$$

If a constraint is convex, the projection is a continuous map. If both constraints are continuous, then a projection algorithm consisting of a combination of the projections is also continuous. If a constraint is "highly continuous", then choice of initialization will have less effect on convergence.

2.12 Conclusions

A model of iterative projection algorithms has been described, dividing the process of convergence into three phases. A statistical model of the progress of convergence is described

and some examples are shown, along with an application of optimizing the performance of the algorithm by restarts. The convergence region is shown to be approximately hyperspherical, with the solution at the centre, although the boundary of the convergence region may be very rough.

The iterative projection algorithm is shown to spiral towards the solution when in the convergence region, with the frequency and damping of the spiral being related to the algorithm parameters and to an "effective angle" between the constraints. Both simple and more complex examples of the spiralling behaviour of the algorithm are shown. A method of speeding up the spiralling convergence process is then described and shown to work well.

Some desirable properties of IPAs were described, and the DM algorithm is shown to possess many of these properties in some form. The effects of uniqueness on algorithm performance and some methods for handling algorithms with more than two constraints have also been described. The commonly used methods of solution estimation and bootstrapping are given a more rigorous basis, followed by a discussion on the effects of noise and possible methods for improving the performance of the algorithm for very noisy data. Finally, there is a section on treating projection algorithms as a map with possible avenues for investigation.

Chapter 3

Reconstruction of Binary Images

3.1 Introduction

The reconstruction of a compact binary image from its Fourier magnitudes is considered here. A primary application of this problem is in macromolecular x-ray crystallography. This particular application is studied in Ch. 4. Protein molecules are reasonably compact structures and it is often useful during the early stages of the structure determination if the boundary, or *molecular envelope*, of the protein molecule can be determined. The molecular envelope can be considered to be a binary image so that the problem addressed applies directly to determining the molecular envelope from crystal x-ray diffraction magnitudes. Note that since crystal x-ray data are considered, the general image reconstruction problem is highly underdetermined, as described in Chapter 1. The objective here is to supplement the problem with binary and compactness constraints to sufficiently constrain the problem.

It is possible to collect diffraction data from appropriately modified crystals that can be processed such that the Fourier magnitudes correspond to those from the envelope alone. Details on the methods of data collection and processing are discussed in the next chapter. The problem then addressed in this chapter is that of using these derived diffraction magnitudes to reconstruct the molecular envelope.

Another application of the binary and Fourier magnitude constraints is in magnetic domain imaging [30]. Unlike proteins, magnetic domain regions tend to be more tenuous, although still approximately connected.

The constraints and projections used to solve this problem are described in the next section. The ability of the constraints to uniquely identify the solution are discussed next, followed by a discussion of the ability of the algorithm to find the true solution. Some experiments in 2-D are used to explore the nature of the problem, followed by conclusions.

3.2 Constraints and projections

Three basic constraints are used in the problem at hand: (1) The Fourier magnitude (data) constraint, (2) the binary constraint, and (3) a connectivity/compactness constraint. Since most IPAs are based on two constraint sets, the image space constraints are combined into a single constraint. Although the application in x-ray crystallography is in 3D, the constraints, projections and results are illustrated here in 2D for convenience. Extension to 3D is obvious and straightforward.

The Fourier magnitude constraint and projection has already been covered in Sec. 1.3.4.6, and the binary and binary with fill fraction constraints and projections have also been covered in Sec. 1.3.4.3.

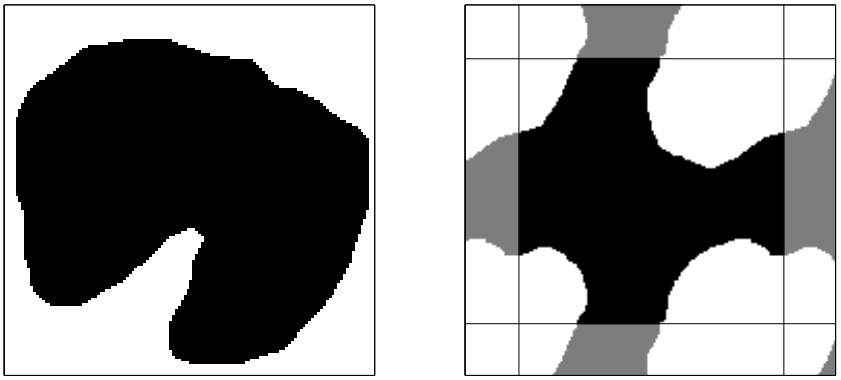
3.2.1 Connectivity constraint

Consider binary images that consist of a single, connected domain (Fig. 3.1(a)), and refer to this as a *connectivity constraint*. Connectivity is in turn defined using a definition of pixel neighbourhoods. A binary image is connected if every 1-pixel is connected to every other 1-pixel through a path of pixel neighbourhoods. Such an image is said to form a single "object". The connectivity constraint set is not easily defined formally in the vector space $\mathbb{R}^{2N}$ unless the space was also endowed with spatial information. While this could be done, it would introduce unnecessary complexity. Connectivity is therefore evaluated in image space.

The significance of connectivity in crystallography is that protein molecules are globular structures that are held together by chemical bonds and non-bonding interactions. The individual molecules, and hence also their envelopes, therefore form a single connected domain. Furthermore, the integrity of the crystal itself is supported by intermolecular contacts so that connectivity also exists between unit cells, as illustrated in Fig. 3.1(b). The image will therefore generally have toroidal symmetry.

Two other related constraints on binary images are also considered. The first is that the image does not contain any "holes", i.e. is *simply connected*, so that any closed contour for a 2D image, or any closed surface for a 3D image, in the object can be shrunk to a point without leaving the object. It is easily seen that this constraint is identical to the connectivity constraint applied to the inverse of the image, i.e. the inverse of a simply connected object is a single object. For periodic images, the non-object part of the image will always constitute at least one "hole", so a simply connected periodic image is defined as having a maximum of one "hole". Although not always the case, most protein molecular envelopes are simply connected.

Consider now the effectiveness of the connectivity constraint as a function of the number of dimensions of the image. The number of neighbouring pixels in a m-dimensional image is $2m$. Since the image must consist of a set of pixels which are all connected together, the



(a) (b)

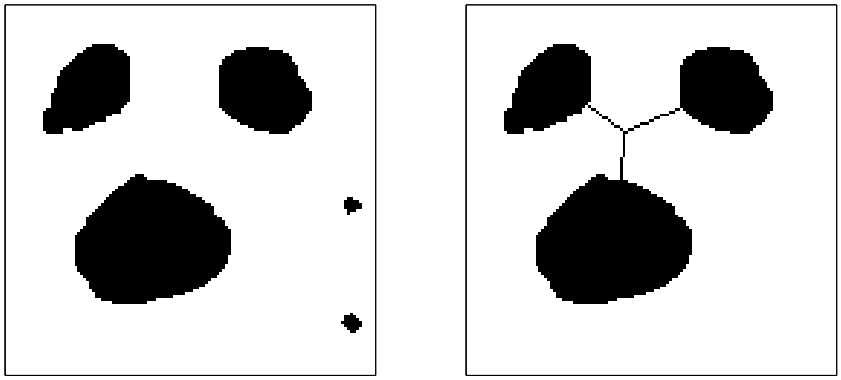
Figure 3.1 (a) A connected image, and (b) an image connected between unit cells.

number of possible connected images increases as the number of neighbours each pixel has increases. So as the number of dimensions of the image increases, the size of the connectivity constraint set increases, making the connectivity constraint less restrictive.

The extreme cases for the dimensionality of the image can be explored by considering an N-pixel periodic binary image where $N = 2^k$ for some integer k. There are 2^N possible binary images. If the image is 1-dimensional, only N^2 images out of the 2^N possible binary images consist of a single connected object. At the other extreme with an k-dimensional image with dimensions $2 \times 2 \times ... \times 2$, all 2^N possible binary images consist of a single connected object since every pixel is connected to every other pixel if a diagonal definition of connectivity is used, e.g. (8-pixel for 2-dimensional images).

The second additional constraint considered is that of the image being "compact". Protein molecules are reasonably compact in the sense that they do not generally form highly tenuous structures. Compactness is not easily defined rigorously, but one could think of, for example, minimizing the moment of inertia of the object subject to the other constraints.

Although difficult to define rigorously, the connectivity constraint is powerful since the size of the constraint set is small. This can be illustrated as follows. If an isolated pixel changes value, due to noise, from 0 to 1 or vice versa, application of the connectivity constraint will change the pixel back to its original value unless the pixel is on the edge of an object. The power of the connectivity constraint to correct noise is therefore in inverse proportion to the number of pixels on the object boundary compared to those inside the object. This is small for large objects in a small number of dimensions, and weakens as the number of dimensions rises since the number of surface pixels increases, further confirming the relationship between the dimensionality of the image and the strength of the connectivity constraint as described above. The connectivity constraint therefore provides good tolerance against noise for a low number of dimensions (as is the case, e.g. 3D, for



(a) (b)

Figure 3.2 A binary image (a) and the result of applying a rigorous connectivity projection (b).

most image reconstruction problems), especially for noise such as additive noise which is spread out over the entire image.

3.2.2 Connectivity projection

The rigorous projection of a binary image onto the set of connected binary images involves adding a network of "filaments" (connected 1-pixels) to connect disconnected pairs of objects whose distances apart are smaller than the area of the smallest of the pair, and removing objects that are smaller than their distance from any other object. This is illustrated in Fig. 3.2. However, although this is the rigorously correct projection, it results in tenuous objects as opposed to the compact objects required, and is therefore not appropriate for the problem at hand.

3.2.3 Compactness projection

The compactness projection with respect to moment of inertia would consist of zeroing pixels starting with those furthermost from the centre of mass, until the desired moment of inertia is reached. If necessary, the zero pixels nearest to the centre of mass can be changed to maintain the overall size of the object. Like the connectivity projection, the compactness projection is not effective, since it is impossible to define a centre of mass for a crystal which is periodic.

3.2.4 A practical combined connectivity and compactness projection

An alternative approach to the rigorous connectivity projection that favours compactness as well as connectivity is to retain only the largest object and delete the remaining objects.

However, in the initial stages, the image will likely consist of many small objects. Furthermore, when the algorithm has partially converged, large objects will tend to be in the vicinity of the true object. An approach found to be effective is to retain objects larger than a threshold size, denoted l, and delete the remaining objects. This method encourages connectivity and compactness, although the projected image is not necessarily connected. The connectivity projection, denoted P_C, is therefore defined by

$$P_C x[\mathbf{t}] = \begin{cases} 1 & \text{if } \mathbf{t} \in L(l) \\ 0 & \text{if } \mathbf{t} \notin L(l), \end{cases} \tag{3.1}$$

where $L(l)$ denotes the set of pixels that belong to objects with area (number of pixels) greater than l. It makes sense for the threshold l to increase with increasing fill fraction, and using $l = \alpha_C f N$, with the constant $\alpha_C \simeq 0.1$, was found to be effective. Application of this projection generally reduces the fill fraction, but after a few iterations, the image tends to contain one or a few large objects and the reduction in fill fraction is rather modest.

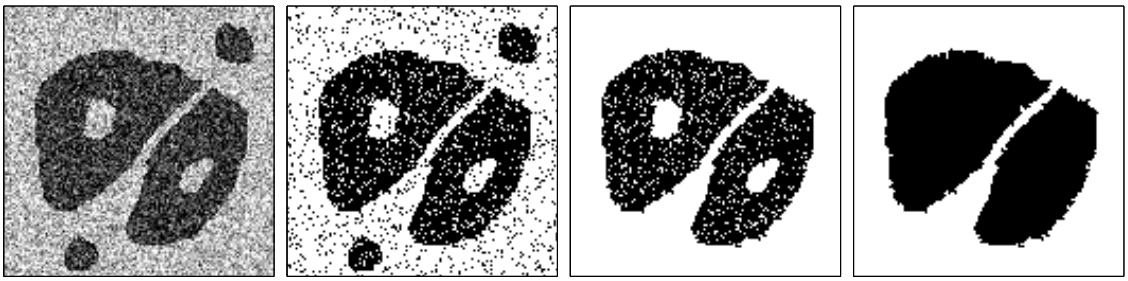
As discussed previously, the image is also required to be simply-connected, i.e. contain no holes. Since the simply-connected constraint is equivalent to the connectivity constraint applied to the inverse of the image, a simply connected projection, denoted P_{SC}, can be implemented in an analogous way to P_C, i.e.

$$P_{SC}\mathbf{x} = 1 - P_C(1 - \mathbf{x}). \tag{3.2}$$

For this projection $l = \alpha_{SC}(1 - f)N$. Note that, as with the connectivity projection not necessarily enforcing a single object, the simple connectivity projection does not necessarily enforce simple connectivity, but allows holes larger than the threshold to remain. This is in fact a useful property since some protein molecules do have large holes. Generally, $\alpha_{SC} = \alpha_C$ has been used, although, for example, α_{SC} could be set to a smaller value if one wanted to allow smaller holes. Note that setting $\alpha_C f N = \alpha_{SC}(1 - f)N$ preserves duality in the sense that an image and its negative will yield identical but inverse results when the algorithm is applied, assuming that the fill fraction is set to $1 - f$ for the negative case.

Since this definition of connectivity ameliorates the problem of a large region becoming disconnected from the main object and erroneously removed, the most restrictive neighbourhood definition can be used, which is a 4-connected neighbourhood for 2D (6-connected for 3D), which encourages compactness as much as possible. Furthermore, since the image is periodic, toroidal connectivity is also allowed.

For most images the order in which the connected and simply connected projections are applied is irrelevant. The fill fraction remains approximately the same, since the connectivity projection reduces the fill fraction and the simple-connectivity projection increases it. It is therefore unnecessary to explicitly enforce the fill fraction constraint during the connectivity constraint, which reduces the computational complexity.



(a) (b) (c) (d)

Figure 3.3 Illustration of application of the image space constraints. (a) $\mathbf{x}$ (b) $P_{BF}\mathbf{x}$ (c) $P_C P_{BF}\mathbf{x}$ (d) $P_I\mathbf{x}$.

3.2.5 Full image space projection

The full binary and connectivity/compactness image space projection P_I is given by

$$P_I\mathbf{x} = P_{SC}P_C P_{BF}\mathbf{x}. \tag{3.3}$$

The result of applying this projection is a binary image that *tends* to be simply connected, has an approximately correct fill fraction, and is reasonably compact. The action of the projection P_I is illustrated in Fig. 3.3. As the algorithm proceeds, the image eventually becomes simply connected.

3.3 Properties

3.3.1 Algorithm properties

The zero-dimensional nature of the image constraint set creates some difficulties. Firstly, the algorithm often takes a large number of iterations with the majority of time spent searching the attractor. This may be the case since the zero-dimensional nature of the constraint means that very little information about the overall Euclidean space geometry can be inferred from the current point, making the search almost a brute force search. This is expected since the binary constraint is a combinatorial-type problem, so it should not be able to be solved quickly and consistently. The attractor search is made even more difficult if the image is large and especially if the low resolution Fourier magnitudes are lost, since that corresponds to a loss of large scale structural information. Secondly, the number of outputs for the binary projection is limited, so if the algorithm is of the type in which the iterate is a projection, such as the ER algorithm, the iterates are more likely get trapped in a limit cycle.

One advantage of the discrete constraint is that convergence is rapid when close to the solution. This can be thought of as the bowl of attraction being small in diameter, but

with steep sides. This property means that if convergence is slow, the problem lends itself well to being run in parallel with different starting points since the convergence problems are mainly due to being unable to find the convergence region of the correct solution as opposed to any local convergence difficulties.

When experimental data are used or noise is added, finding the solution becomes more difficult for all problems. The difficulty is exacerbated for this particular problem for two reasons. Firstly, if the low resolution Fourier magnitudes are lost, the higher resolution Fourier magnitudes generally have low magnitudes, which can quickly be overwhelmed by any noise in the data, so the true solution is no longer a fixed point, and there may be many false solutions which also create regions of attraction which can trap the iterate. Secondly, the zero dimensional binary constraint set means that there is unlikely to be a true solution which satisfies all the constraints, which may make the algorithm unstable even when at the solution.

If the true solution is found the estimate is within the noise limit of the true solution since the steps in the algorithm are proportional to the distance between constraints. If an image domain projection is used as the final estimate, the estimate can often be significantly better due to the power of the discrete binary connectivity constraint.

3.3.2 Uniqueness

The zero-dimensional nature of the binary and connectivity constraint means that the probability of an intersection between the Fourier magnitude constraint space manifold and the image domain constraint set is infinitesimally small. So the presence of a false image which satisfies the constraints is very unlikely. However, there can be images which satisfy the constraints that are not the true image, but are *trivially related* to the true image.

Firstly, the constraints are unable to distinguish a circularly shifted image from the true image. Secondly, the inverted image created by flipping an image in the origin will also satisfy all the constraints. Thirdly, without application of the fill fraction constraint, the constraints are unable to distinguish the inverse image from the true image. If the fill fraction is exactly $f = 0.5$, then the image and its inverse cannot be distinguished.

3.3.3 Attraction to the negative of the solution

There are ${}^{N}C_{fN}$ possible N-pixel binary images with fill fraction f, and this number is maximized if $f = 0.5$. Disregarding the connectivity constraint, convergence is expected to be the most difficult when f is close to 0.5, i.e. when the constraint set is largest. Furthermore, if the fill fraction constraint is applied and f is close to, but not equal, to 0.5, the algorithm may become trapped near the negative of the image, since this image is similar to the true solution, satisfying all the constraints except for the fill fraction constraint. This situation does not occur if $f = 0.5$ exactly since the algorithm can simply converge to the negative image.

A method used to escape from the above false solution is as follows. The algorithm is run for I_{long} iterations and then both the iterate, $\mathbf{x}$, and the negative of the iterate, $1 - \mathbf{x}$, are calculated and used as the starting iterate to run the algorithm for a further I_{short} iterations each. The iterate with the lower error metric is selected, and another I_{long} iterations are run, and so forth. An example of a reconstruction of a 128×128 pixel binary image with Fourier magnitude and binary, fill fraction and connectivity constraints with $f = 0.6$ is shown in Fig. 3.4, for which $I_{long} = 300$ was used. Inspection of the figure shows that the iterate is initially trapped near a local minimum. When the iterate is negated (inverted) the true solution is found in around 40 iterations.

The value of I_{short} should be chosen to be greater than the number of iterations required for convergence, and I_{long} should be small enough that the iterate will not move out of the bowl of attraction of the negative image before the negation operation is performed. If the search of the attractor is considered to be a constant probability process as described in Sec. 2.2, then searching at the negative solution is still searching the same attractor but with a different initial image, and is therefore not a waste of iterations. Since the error metric falls near-monotonically during convergence, if the algorithm has converged or is in the convergence phase on either of the branches, the selection of the branch with the lower error metric will allow the algorithm to continue its convergence.

3.3.4 Image domain error metrics

A list of common error metrics has already been presented in Sec. 1.4. As described in that section, a definition of the image domain error metric needs to take the trivially related solutions into account. An efficient method of accomplishing this for the specific case of binary and Fourier magnitude constraints is described here.

If the correct and reconstructed images are registered correctly, the number of pixels in error E_n for a periodic binary image is

$$E_n = \sum_{\mathbf{t}} |\hat{\mathbf{x}}[\mathbf{t}] - \mathbf{x}[\mathbf{t}]| \tag{3.4}$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the correct and reconstructed images respectively and the summation is over all pixels in the image. If the images are not correctly registered, then the reconstructed image must be circularly shifted over all possible positions, the error calculated, and the true error is then the minimum. This can be done efficiently by using a circular correlation which calculates the agreement between the two images for different shifts. For each shift $\mathbf{d}$, the number of pixels in error is

$$E_n[\mathbf{d}] = \sum_{\mathbf{t}} |\hat{\mathbf{x}}[\mathbf{t}] - \mathbf{x}[\mathbf{t}]| \tag{3.5}$$

$$= \sum_{\mathbf{t}} \mathbf{x}[\mathbf{t}] + \sum_{\mathbf{t}} \hat{\mathbf{x}}[\mathbf{t}] - 2(\hat{\mathbf{x}} \odot \mathbf{x})[\mathbf{d}] \tag{3.6}$$

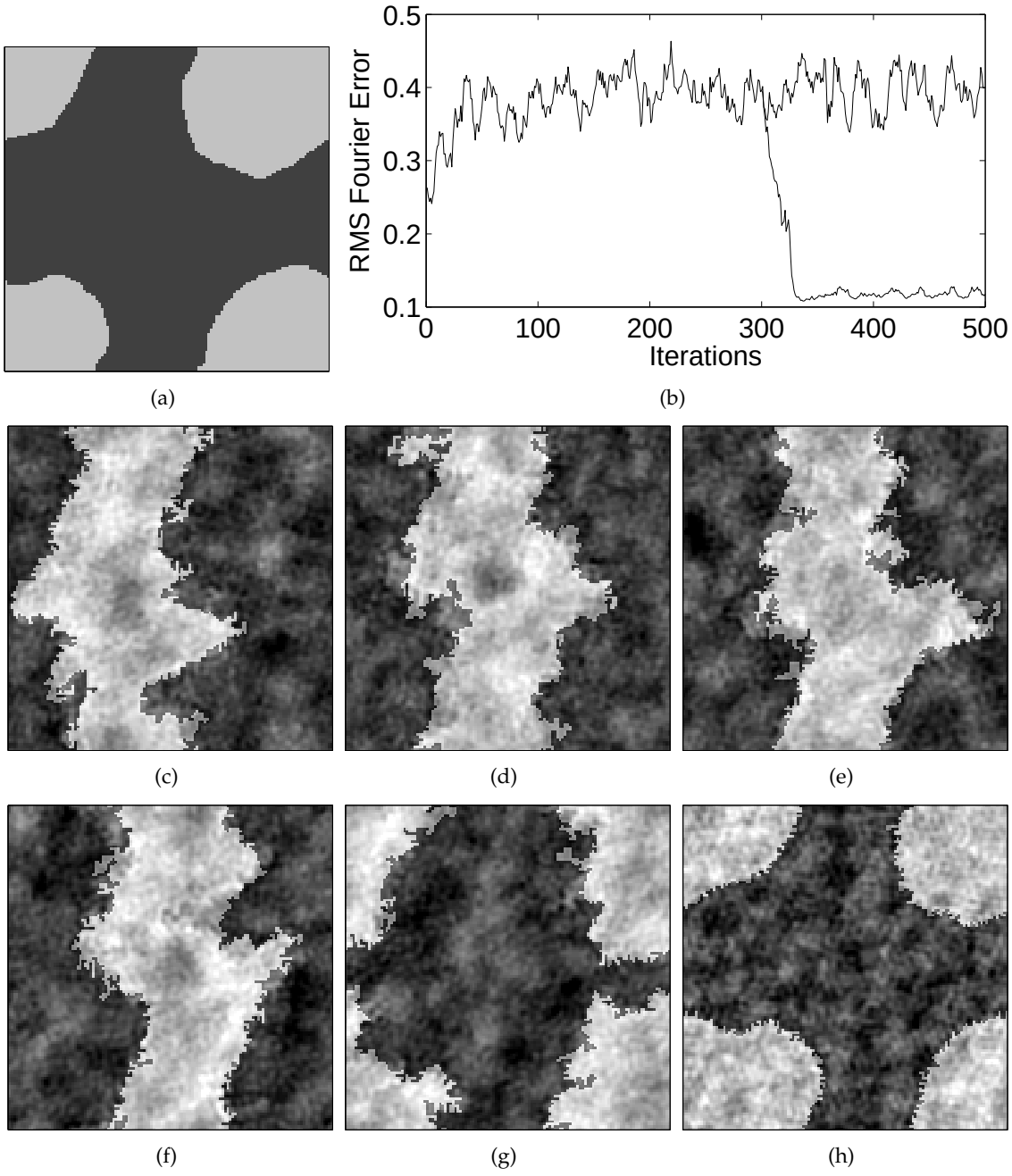


Figure 3.4 Example where the method described in Sec. 3.3.3 finds the solution. (a) True image, (b) RMS Fourier magnitude error versus iteration, (c) iiterate at 100 iterations, (d) Iterate at 290 iterations, (e) iterate at 310 iterations without negation operation, (f) iterate at 350 iterations without negation operation, (g) iterate at 310 iterations with negation operation after 300 iterations, and (h) iterate at 350 iterations with negation operation after 300 iterations.

where $(\hat{\mathbf{x}} \odot \mathbf{x})[\mathbf{d}]$ is the value of the circular correlation at $\mathbf{d}$. Since the first two terms in Eq. (3.6) are constant, the shift $\mathbf{d}$ which minimizes E_n is when $\hat{\mathbf{x}} \odot \mathbf{x}$ is maximized, and so

$$E'_n = \sum_{\mathbf{t}} \mathbf{x}[\mathbf{t}] + \sum_{\mathbf{t}} \hat{\mathbf{x}}[\mathbf{t}] - 2\max(\hat{\mathbf{x}} \odot \mathbf{x}) \tag{3.7}$$

where $\max(\cdot)$ is the maximum value of the image. Similarly, the error between the negative of the reconstructed image $1 - \hat{\mathbf{x}}$ and the correct image is given by

$$E''_n = N - \sum_{\mathbf{t}} \mathbf{x}[\mathbf{t}] - \sum_{\mathbf{t}} \hat{\mathbf{x}}[\mathbf{t}] + 2\min(\hat{\mathbf{x}} \odot \mathbf{x}).$$

Similarly, for the version of the estimated image inverted in the origin,

$$E'''_n = \sum_{\mathbf{t}} \mathbf{x}[\mathbf{t}] + \sum_{\mathbf{t}} \hat{\mathbf{x}}[\mathbf{t}] - 2\max(\hat{\mathbf{x}'} \odot \mathbf{x}) \tag{3.8}$$

$$E''''_n = N - \sum_{\mathbf{t}} \mathbf{x}[\mathbf{t}] - \sum_{\mathbf{t}} \hat{\mathbf{x}}[\mathbf{t}] + 2\min(\hat{\mathbf{x}'} \odot \mathbf{x}) \tag{3.9}$$

where $\hat{\mathbf{x}'}$ denotes $\hat{\mathbf{x}}$ inverted in the origin, i.e. $\hat{\mathbf{x}'}[\mathbf{t}] = \hat{\mathbf{x}}[-\mathbf{t}]$.

The minima of all the image errors given by $\min(E'_n, E''_n, E'''_n, E''''_n)$ is then the true image error.

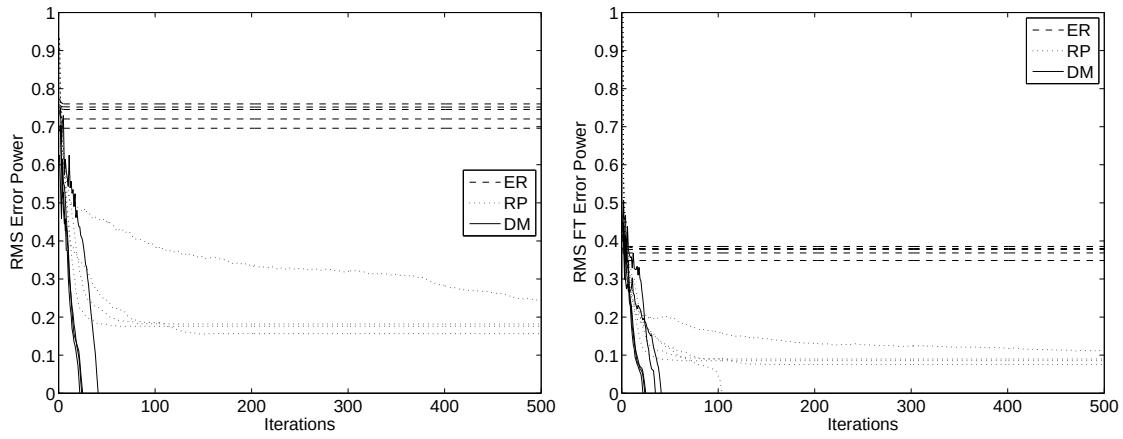
3.4 Simulations

Experiments illustrating various aspects of an IPA using the projections described in Sec. 3.2 were carried out with 2D images using various combinations of the image domain constraints. The 2D image is a 128×128 pixel image with $f = 0.4$, with a single simply connected region as shown in Fig. 3.6(c). Fourier magnitudes are available in a circle with diameter 128 centred on the zero frequency term. Different fill fractions and grid sizes were created by scaling the image. Note that for any fill fraction, the dual problem with the pixels flipped gives identical results as long as the initialization point is also flipped around the 0.5 value.

3.4.1 Algorithm choice

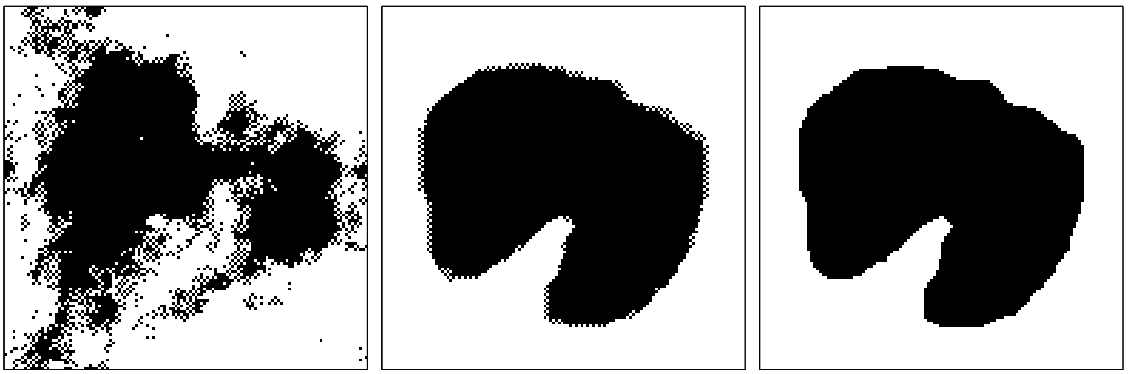
The ER, RP and DM algorithms were investigated first. Five different starting images were used for each of the ER, RP ($\gamma = 0.7$), and DM ($\beta = 0.7$) algorithms. The RMS image error and RMS Fourier errors are shown in Fig. 3.5(a) and Fig. 3.5(b) respectively. One of the estimated solutions from each of the three algorithms is shown in Fig. 3.6.

As expected with a highly non-convex constraint, the ER algorithm stagnated almost immediately and did not make any progress towards a solution. The RP algorithm performed



(a) (b)

Figure 3.5 Error metrics for 5 runs using the ER, RP, and DM algorithms (a) Image Error (b) Fourier Error.



(a) (b) (c)

Figure 3.6 Final estimates of three runs from Fig. 3.5. (a) ER algorithm (b) RP algorithm (c) DM algorithm. The DM algorithm reconstructs the image perfectly.

better with its ability to avoid stagnation and was able to find the convergence region of the true solution. However, its poor convergence properties did not allow it to find the true solution. Furthermore, in some other cases which are not shown here, the RP algorithm gets trapped in a 2-iteration limit cycle due to its simplicity. The DM algorithm performed the best with the fastest convergence and with perfect reconstruction of the image in all five runs. Using different missing data, choice of parameters, or image domain constraints did not affect conclusions, with the DM algorithm generally outperforming the RP algorithm which in turn outperforms the ER algorithm. The DM algorithm is therefore used exclusively in the following sections.

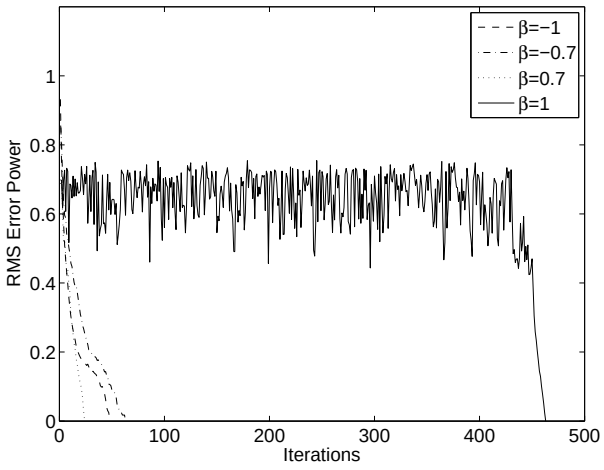


Figure 3.7 RMS image error vs iterations for different choices of β.

3.4.2 DM algorithm choice

The effects of different parameter choices on the DM algorithm are explored. The DM equation given in Eq. (1.67) is recalled to be

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \beta[P_I T_M \mathbf{x} - P_M T_I \mathbf{x}], \tag{3.10}$$

with P_I and T_I denoting the projection and relaxed projection of the image domain constraint, and P_M and T_M denoting the corresponding Fourier magnitude projections. The parameters β, γ_I and γ_M are as defined for Eq. (1.67). Setting $\gamma_I = -1/\beta$, $\gamma_M = 1/\beta$, Fig. 3.7 shows typical image error plots for $\beta = -1, -0.7, 0.7, 1$. The first three choices of β perform well, but setting β to 1 did not work well, with the convergence region appearing to be quite small, so the algorithm spends a long time searching the attractor. Setting $\gamma_I = -1$, $\gamma_M = 1/0.7$, $\beta = 0.7$ did not work well either, suggesting that setting the parameter $\gamma_I = -1$ is a problem. The choice of $\beta = 0.7$ performed better in the convergence phase than for negative values of β. This may be due to the fact that the algorithm can be seen as driving the iterate towards $P_I T_M \mathbf{x}$, which is an excellent estimate of the solution when the iterate is near to the solution due to the "$2F_o - F_c$" effect described in Sec. 2.8.3 if γ_M is close to 1. Additionally, an excellent estimate of the solution at each iteration is given by $P_I T_M \mathbf{x}$. Values of $\beta = 0.7 - 0.9$ are therefore used even though they require twice as many iterations as when $|\beta| = 1$.

3.4.3 Fill fraction dependence

Different fill fractions were achieved by scaling the image and 50 runs were made (with different starting images) for each fill fraction with a maximum of 500 iterations per run

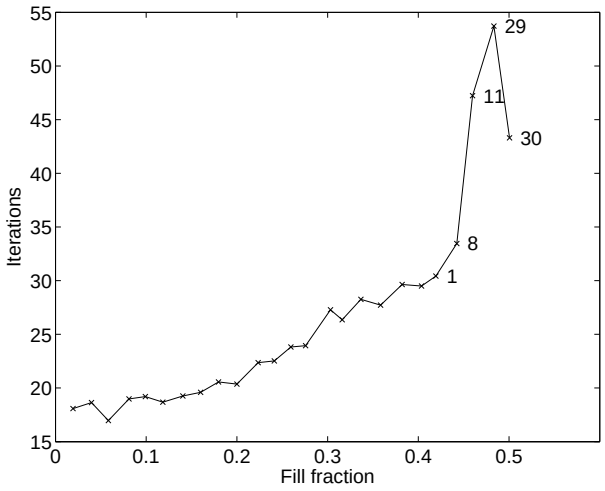


Figure 3.8 Mean number of iterations needed for convergence vs fill fraction. Where applicable, the number of non-converged runs out of 50 is shown.

and $\beta = 0.7$. A plot of the mean number of iterations needed for convergence versus the fill fraction is shown in Fig. 3.8. Convergence is defined as when the algorithm perfectly reconstructs the true solution. Where applicable, the number of non-converged runs out of 50 is also shown. The number of iterations required increases as the fill fraction increases, and the number of failures also increases, as anticipated. Note again that f behaves identically to $1 - f$. The number of iterations required for convergence falls when $f = 0.5$ since the algorithm can converge to either the true image or the inverse of the true image.

3.4.4 Missing data

To test the effect on the algorithm of missing low order Fourier magnitude data, the four lowest order Fourier magnitude data were removed in addition to the high resolution Fourier data as described above. Ten runs were made for each combination of constraints with a maximum of 10000 iterations with $\beta = 0.7$ and $f = 0.4$. The results are shown in Table 3.1.

The fill fraction information can be enforced in either the image domain using the P_{BF} projection or in the Fourier domain by setting the origin value. Case 2 performed required significantly more iterations than cases 1, 4, and 5, which performed similarly. This suggests that having the fill fraction information available is important and that the algorithm performs approximately the same regardless of how the fill fraction constraint is enforced. Enforcing the fill fraction in the image domain is computationally more intensive, but has the advantage in practical situations where the levels for the binary constraint are not known, as will be described in Chapter 4. For this reason, the fill fraction will be enforced in the image domain.

Table 3.1 Performance for various combinations of the image and Fourier domain constraints.

Case	Image domain constraint	Fourier Magnitude data	Number successful	Iterations required
1	P_{Bin}	With origin	10	21-42
2	P_{Bin}	Without origin	10	28-257
3	P_{Bin}	Without Low Res	0	N/A
4	P_{BF}	With origin	10	18-39
5	P_{BF}	Without origin	10	18-39
6	P_{BF}	Without Low Res	0	N/A
7	$P_{SC}P_CP_{BF}$	With origin	10	15-30
8	$P_{SC}P_CP_{BF}$	Without origin	10	16-30
9	$P_{SC}P_CP_{BF}$	Without Low Res	7	22-56

The algorithm behaved identically for Cases 4 and 5. This is expected since if $\mathbf{x}$ has the correct fill fraction, then the output of the Fourier magnitude projection $P_F\mathbf{x}$ will also have the correct fill fraction, regardless of whether or not the zero frequency term is available.

Like most images, much of the energy in the Fourier magnitudes is in the lower resolution samples. This effect is increased for the binary connected images used which consist of large globular objects. The low resolution data is therefore extremely important as their large values correspond to a large amount of information stored in each value. Only the 4 Fourier samples immediately surrounding the zero frequency value were removed, but there was a drastic decrease in the number of successful runs and an increase in the number of iterations required (Table 3.1). The connectivity/compactness constraint was needed to reconstruct the correct solution. Note that in all 9 cases, an additional 3535 high frequency values out of the 16384 Fourier values have already been removed, but these did not drastically affect results. The higher magnitudes are kept reasonably large by the sharp edges between the regions. Although they do not store much direct structural information, they provide a useful check for uniqueness. Thus without the low resolution information, convergence is much more difficult, but uniqueness is still preserved. This can be seen by the Fourier magnitude errors remaining large for failed reconstructions.

3.4.5 Effect of noise

There will always be noise in experimental data, so the Fourier magnitudes will not be those of a perfect binary image. Two sources of noise are considered here. The first is the noise generated during the collection of the Fourier magnitude data, which is modeled by adding white Gaussian noise to the entire image. The second occurs when the solvent contrast data is generated, where the edges of the envelope will not be perfectly sharp.

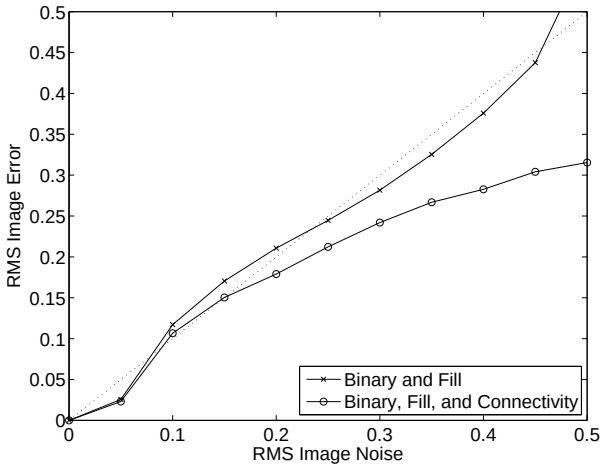


Figure 3.9 RMS noise in data vs RMS noise in reconstruction. The dotted line represents equality of the input and output noise power.

This represents a form of noise localized on the boundaries of the envelope.

3.4.5.1 Gaussian noise

Gaussian noise was added to the image, and the Fourier magnitudes calculated. This is equivalent to adding Rayleigh noise to the Fourier magnitudes. The effect of the noise was investigated by using a 128×128 pixel image with $f = 0.4$. Fifty runs were made for each noise level, and the mean output error versus the noise RMS level is plotted in Fig. 3.9.

The Gaussian noise is distributed evenly over the entire image, and the discrete constraints mean that the error in the reconstructed image is often less than the noise level in the data. Furthermore, since the Gaussian noise is distributed over the whole of the image, the connectivity projected image is essentially immune to noise except at the boundaries of the objects, so the noise sensitivity is reduced.

A series of images from the runs in Fig. 3.9 are shown in Fig. 3.10, one using a binary and fill fraction constraint, and the other using the binary, fill fraction, and connectivity constraint. The same starting image and noise of 0.3 RMS was used. The strength of the image domain constraints can be seen by applying the projections to the noisy images as shown in Fig. 3.10(c) and Fig. 3.10(e). Despite the high level of noise, the reconstructions were successful as shown in Fig. 3.10(d) and Fig. 3.10(f).

The RMS image error power and the RMS Fourier error power versus iteration are shown in Fig. 3.10(g) and Fig. 3.10(h) respectively. As expected, the reconstruction using the connectivity constraint has a lower image error. However, its Fourier error is higher since the small constraint set size of the connectivity constraint means that the reconstruction is biased towards the image domain constraint, and the Fourier domain error suffers as a result. This is not a problem, to the contrary, since the Fourier error is a measure of conver-

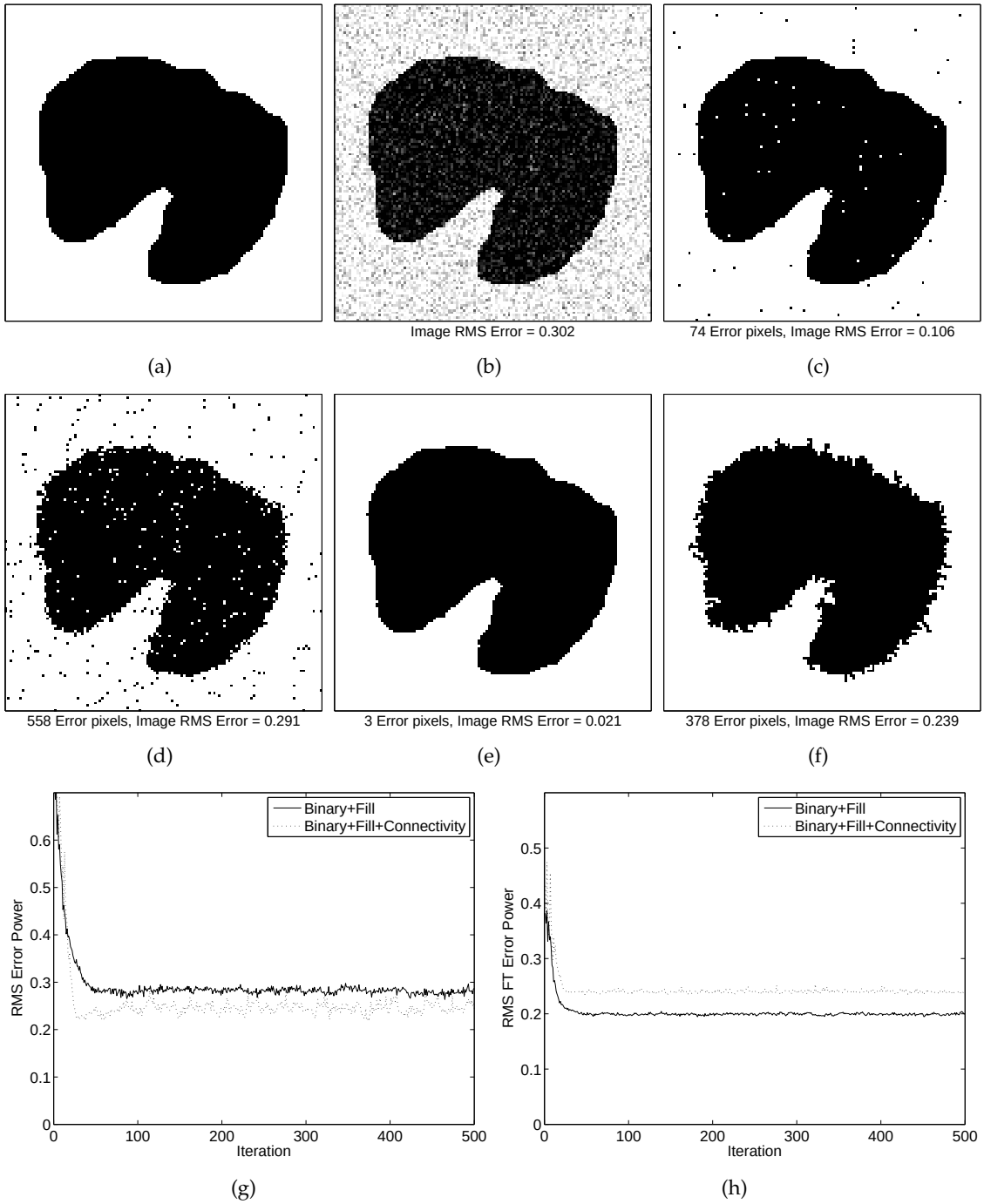


Figure 3.10 Effect of noise on algorithm performance. (a) Correct image, (b) Noisy image, (c) Binary and fill fraction projection applied to (b). (d) Reconstruction using the binary and fill fraction constraint, (e) Binary, fill fraction and connectivity projection applied to (b). (f) Reconstruction using the binary, fill fraction and connectivity constraint. (g) RMS image error vs Iteration and (h) RMS Fourier error vs iteration.

gence, the greater distance from convergence suggests that further iterations may be able to drive the error down further. Note that finding an estimate with a good image error is generally considered more important than finding an estimate with a good Fourier error. If the Fourier error was important, then $P_F\mathbf{x}$ can be used as the estimate instead of $P_I T_F \mathbf{x}$.

3.4.5.2 Edge noise

After processing solvent contrast data in a crystallography experiment the edges of the implied molecular envelope are often not perfectly sharp, and can be modeled by convolving the binary image with a Gaussian function. The effect of this is now investigated. A Gaussian was convolved with the binary image shown in Fig. 3.11(a) to create the image shown in Fig. 3.11(b) which was then used to create the Fourier magnitude data. A cross section of Fig. 3.11(b) through the middle is shown in Fig. 3.11(c) which shows the Gaussian roll-off. The roll-off can be considered as a form of localized noise. Due to the distribution of the Gaussian function, it was difficult to create data with 0.3 RMS noise while retaining a realistic level of roll-off, so the total noise power was set to 0.17 RMS. The reconstruction algorithm was then run using this modified data. One aspect of the Gaussian roll-off is that the noise rarely causes the value of the pixels to be closer to the other binary value, so the image domain projection cleans up the noise well as shown in Fig. 3.11(d) and Fig. 3.11(f).

The algorithm with the connectivity constraint is relatively more affected than the Binary+Fill constraint case. This is to be expected since the binary and fill constraint is not affected by the edges, and in fact the reconstruction in Fig. 3.11(e) seems to have found a solution which is approximately correct when averaged over a small neighbourhood of pixels. The localization of the noise in the reconstructions can be seen in Figs. 3.11(e) and 3.11(g).

3.4.5.3 Edge noise correction

The envelope can be sharpened to correct the rounded edges. If the edges are Gaussian rounded, then in principle the envelope can be recovered by multiplying the Fourier magnitudes with the inverse of a Gaussian, i.e.

$$X(\mathbf{h}) = \frac{1}{G(\mathbf{h})} M(\mathbf{h}) \tag{3.11}$$

where $M(\mathbf{h})$ is the Fourier magnitude data, $X(\mathbf{h})$ are the Fourier magnitudes of the binary envelope, and $G(\mathbf{h}) = \mathcal{F}[g(\mathbf{h})]$ is the Fourier transform of the smoothing Gaussian. Note that the phase of $G(\mathbf{h})$ is always zero if $g(\mathbf{h})$ is Hermitian, i.e. an even function for real $\mathbf{g}$.

In a real problem there is noise, and if the power spectral density of the noise can be estimated, Wiener deconvolution should be used instead. The Wiener deconvolution minimizes the RMS distance of the deconvolved envelope from the true binary envelope and is

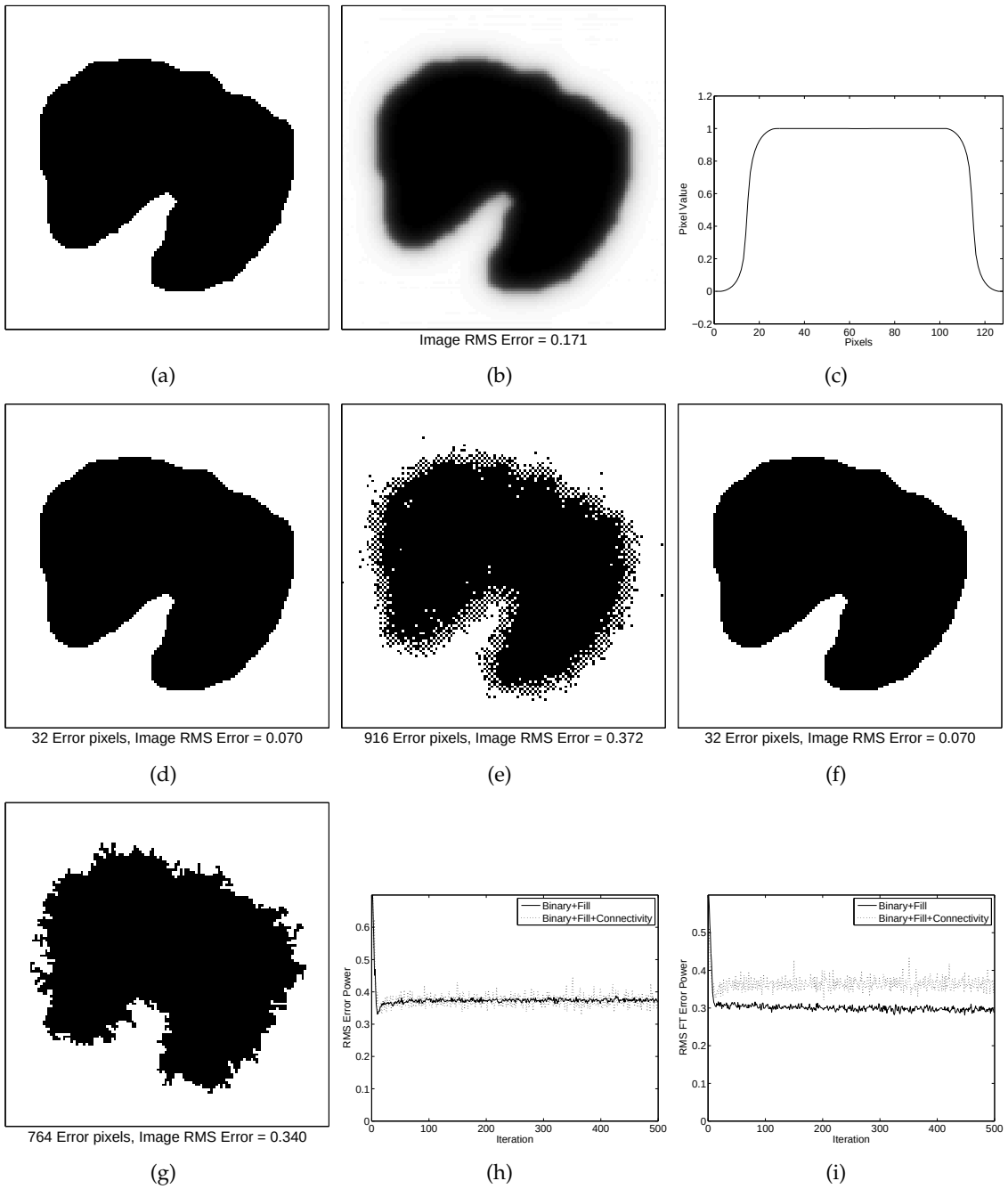


Figure 3.11 Effect of noise on algorithm performance. (a) Correct Image, (b) Noisy Image. (c) Horizontal cross section through the middle of (b) showing the edge profile. (d) Binary+Fill projection applied to (b). (e) Reconstruction using Binary+Fill constraint. (f) Binary+Fill+Connectivity projection applied to (b). (g) Reconstruction using Binary+Fill+Connectivity constraint. (h) RMS Image Error vs Iteration and (i) RMS Fourier error vs Iteration.

given by

$$X(\mathbf{h}) = \frac{G(\mathbf{h})M(\mathbf{h})}{|G(\mathbf{h})|^2 M(\mathbf{h}) + N(\mathbf{h})} M(\mathbf{h}) \tag{3.12}$$

where $N(\mathbf{h})$ is the mean power spectral density of the noise. Note any imperfection in the deconvolution will show up as noise which corresponds to distance between constraints in $\mathbb{R}^{2N}$.

3.5 Conclusions

Reconstruction of compact binary images from undersampled Fourier magnitude data maps to the molecular envelope determination problem in x-ray crystallography. IPAs for this problem were developed incorporating binary, connectivity, and compactness constraints, and formulating appropriate projection operators using the ER, RP and DM algorithms. The DM algorithm performed the best, and the RP algorithm worked in some cases, finding the solution but suffering from poor final convergence. The discrete nature of the image domain constraints causes the ER algorithm to stagnate very quickly. The binary constraint on the image renders the problem unique, but additional constraints are needed to find the solution if there is further data loss, particularly at low spatial frequencies. The image domain constraints are shown to be robust to some types of noise expected in realistically noisy data, and correction techniques to mitigate the remaining types of noise to which the image domain constraints are less robust are suggested. The DM algorithm is shown to be effective with the realistically limited and noisy data, and is potentially useful for the crystallographic problem. In the next chapter, the techniques developed in this chapter are applied to the reconstruction of real protein envelopes.

Chapter 4

Reconstruction of Molecular Envelopes

4.1 Introduction

Application of structural constraints in crystallographic phase retrieval requires knowledge of the region occupied by the molecule, the so-called molecular envelope. However, the molecular envelope is usually determined from preliminary electron density functions, calculated using experimentally-derived phases, and so *ab initio* envelope determination presents a catch-22 situation. Solution scattering [31, 32, 33, 34, 35, 36, 37] or electron microscopy [38, 34, 39, 40] can be used to derive molecular envelopes, however it would be useful if the molecular envelope could be determined directly from the structure factor magnitudes of the crystal. One approach which potentially allows this is the method of contrast variation, which can be used to obtain estimates of the magnitudes that would be diffracted by the molecular envelope itself. The problem then reduces to phasing these derived magnitudes.

Solvent contrast variation involves the collection and analysis of diffraction data from macromolecular crystals, where the scattering contribution from the bulk solvent has been systematically varied. The potential uses of such information have long been known. By manipulating the electron density of the solvent, Bragg and Perutz (1952) [41] were able to observe systematic changes in the intensity of the low order diffraction data collected from Haemoglobin crystals, and infer the approximate dimensions of the molecule. A number of contrast variation experiments have subsequently been used to estimate molecular envelopes, usually by changing the salt or the salt concentration [42]. Another way of modulating diffraction from the bulk solvent is to disperse anomalous scatterers in it and make diffraction measurements about an absorption edge [8, 43, 44]. An advantage of the latter approach is that isomorphism is conserved. Whichever method is used, the result, ideally, is the extraction of structure factor magnitudes due to the molecular envelope alone, i.e. a function equal to unity within the envelope and zero outside. This function is sometimes referred to as the indicator function of the envelope, although it will be referred to here simply as the envelope.

Once estimates of the structure factor magnitudes have been obtained, the problem is to phase these magnitudes to obtain the molecular envelope itself. As pointed out by Shepard *et al.* (2000) [44], this phasing problem has a quite different character to the usual phase problems in crystallography. The corresponding electron density is not atomistic, it does not have the detailed structure of a low resolution protein electron density, the electron density is far from being randomly distributed in the unit cell but is a rather compact binary function, and the number of (low resolution) structure factor magnitudes that are used as data is quite small. However, both Carter *et al.* (1990) and Fourme *et al.* (1995) [42, 43] argued that the problem has some similarities to small molecule structure determination and used methods based on direct methods to phase the envelope diffraction magnitudes. Carter *et al.* used solvent contrast variation data and direct methods phasing to determine an 18Å resolution envelope of tryptophanyl-tRNA synthase from *Bacillus stearothermophilus*. Results were promising; however, they had the advantage of 6-fold non-crystallographic symmetry at low resolution, and the method required considerable manual intervention. Fourme *et al.* showed that measurable anomalous scattering solvent contrast measurements could be made for two proteins, although there were experimental difficulties and the data were not used for envelope determination. They noted that the potential of the method for complex structures depends critically on the initial phase determination of the envelope magnitudes by direct methods, which has not yet been convincingly demonstrated. Shepard *et al.* (2000) [44] took a different approach and represented the envelope as a surface in spherical polar coordinates, and parameterised the surface using spherical harmonics and a small number of coefficients. The coefficients are determined from the envelope structure factor magnitudes using a nonlinear least-squares minimization procedure. Encouraging results were obtained using simulated data, although they noted that their method cannot represent general envelopes (since the actual surface function may be multi-valued), and there were difficulties with scaling the data and robustness of the gradient-based minimization procedure.

Neutron diffraction has also been explored for envelope determination by using differing hydrogen/deuterium contents to vary the solvent scattering. Badger (1996) [45] used solvent contrast neutron diffraction data from cubic insulin crystals, and application of a search procedure with a cost function that favours a binary histogram, to estimate the molecular envelope. However, the method is suitable only for the centric reflections and the search procedure is not feasible for a large data set.

The preceding chapter showed that the characteristics of molecular envelopes should allow a unique reconstruction from the structure factor magnitudes alone. The techniques developed there are now applied to the reconstruction of a real protein envelope, making appropriate adjustment for experimental realities. In the next section contrast variation methods for deriving molecular envelope structure factor magnitudes are briefly reviewed. The changes in the algorithm which need to be made for protein envelopes are described in Sec. 4.3, and the results of simulations for two protein crystals are presented in Sec. 4.4. Concluding remarks are made in the final section.

4.2 Envelope structure factor magnitudes from solvent contrast variation data

The use of either solvents with different electron densities or solvents containing anomalous scatterers to derive the structure factor magnitudes of the molecular envelope has been described elsewhere [42, 43], but the key elements of these calculations are briefly outlined here.

Consider a unit cell with a protein molecule surrounded by solvent with electron density ρ_s. Let the binary envelope function be denoted $x_b(\mathbf{t})$, where $\mathbf{t}$ is position in real space. The electron density in the unit cell, $x_e(\mathbf{t})$, can then be written as

$$x_e(\mathbf{t}) = x_p(\mathbf{t}) + \rho_s(1 - x_b(\mathbf{t})), \tag{4.1}$$

where $x_p(\mathbf{t})$ is the electron density of the protein alone. The structure factors (Fourier values) are then

$$X_e[\mathbf{h}] = X_p[\mathbf{h}] - \rho_s X_b[\mathbf{h}] \quad , \quad \mathbf{h} \neq \mathbf{0}, \tag{4.2}$$

where $\mathbf{h} \neq \mathbf{0}$ denotes all structure factors other than the origin term. The equation for $\mathbf{h} = \mathbf{0}$ is somewhat different but is of little significance since $X[\mathbf{0}]$ cannot be measured. Straightforward manipulation of Eq. 4.2 shows that the measured magnitudes are given by

$$|X_e[\mathbf{h}]|^2 = |X_p[\mathbf{h}]|^2 + \rho_s^2|X_b[\mathbf{h}]|^2 - 2\rho_s Re\{X_p[\mathbf{h}]X_b^*[\mathbf{h}]\} \quad , \quad \mathbf{h} \neq \mathbf{0}, \tag{4.3}$$

Equation 4.3 is linear in the three unknowns $|X_p[\mathbf{h}]|^2, |X_b[\mathbf{h}]|^2$, and $Re\{X_e[\mathbf{h}]X_b^*[\mathbf{h}]\}$, so if data are collected for three different solvent electron densities ρ_s, then the three corresponding equations can be solved for these unknowns. In particular, the structure factor magnitudes of the molecular envelope, $|X_b[\mathbf{h}]|$, can be obtained. In practice the three data sets need to be put onto a common scale. Also, some means of making the boundary between protein and solvent more step-like must be introduced. However the description above shows the essence of the technique.

An alternative means of manipulating the scattering from the bulk solvent is to incorporate anomalous scatterers in the solvent, and make measurements at different wavelengths. Advantages of this approach are that only a single crystal is required and there is no lack of isomorphism. The structure factors at wavelength λ are then given by

$$X_e[\mathbf{h}, \lambda] = X_p[\mathbf{h}] - [\rho_s + aK(\lambda)]X_b[\mathbf{h}] \quad , \quad \mathbf{h} \neq \mathbf{0}, \tag{4.4}$$

where $K(\lambda)$ is the known, complex, wavelength-dependent, scattering by the anomalous scatterers and a is a constant related to the concentration of anomalous scatterers in the solvent. Manipulation of Eq. 4.4 shows that the measured magnitudes are given by

$$|X_e[\mathbf{h}, \lambda])|^2 = |X_p[\mathbf{h}]|^2 + |\rho_s + aK(\lambda)|^2|X_b[\mathbf{h}]|^2 - 2Re\{X_p[\mathbf{h}](\rho_s + aK(\lambda))^* X_b^*[\mathbf{h}]\} \quad , \quad h \neq 0. \tag{4.5}$$

Hence, similarly to the previous case, measurements of $|X_e[\mathbf{h}, \lambda]|$ for different wavelengths (but fixed ρ_s) gives a system of linear equations that can be solved for $|X_b[\mathbf{h}]|$. Since $K(\lambda)$ is complex, and so $X_e[\mathbf{h}, \lambda] \neq X_e[-\mathbf{h}, \lambda]$, two equations are obtained for each wavelength, and data for two wavelengths are in principal sufficient to solve for $|X_b[\mathbf{h}]|$. In practice, the methods of multiple anomalous dispersion (MAD) [46] could be used to obtain a stable solution of Eq. 4.5.

4.3 Algorithm

The problem at hand is to reconstruct the molecular envelope $x_b[\mathbf{t}]$ from its structure factor magnitudes $|X_b[\mathbf{h}]|$. The method used to reconstruct the envelope is based on the algorithm described in Chapter 3 to reconstruct a binary image from undersampled Fourier magnitude data. The Fourier domain constraint consists of a Fourier magnitude data constraint with some missing data values and an unknown scale factor. The image domain constraint consists of binary, fill fraction and connectivity constraints as described in Chapter 3.

A number of practical issues are first discussed. The effect of missing low resolution Fourier magnitude data is explored in Sec. 4.4.2, and determination of the scale factor in Sec. 4.3.2. The effect of crystallographic symmetry on the reconstruction problem, focussing on the common $P2_12_12_1$ crystallographic space group symmetry, is discussed in Sec. 4.3.3.

4.3.1 Missing Fourier magnitude data

The molecular envelope is a low resolution object and so only the low resolution magnitudes, say less than $7 - 10$Å resolution, are pertinent. However, an important practical consideration is that the Fourier magnitude data are available only down to a minimum resolution of, say, 50Å. The lower resolution limit presents a particular difficulty in this problem as described in the previous chapter. It is assumed that magnitude data $M[\mathbf{h}]$ (the measured value of $|X_b[\mathbf{h}]|$) have been obtained, subject to the usual errors, from some kind of solvent contrast variation experiment as described above, and that only the data between the resolutions d_{min} and d_{max} are available. The missing data can be a significant fraction of the total amount of energy in the Fourier magnitudes, especially for globular binary images which have a lot of energy in the lower resolution Fourier magnitudes.

4.3.2 Scale factor

The Fourier magnitude data are measured on an arbitrary scale and it is therefore not possible to set the values of the binary function. However, any two-valued function with values a and b with $a < b$ can be transformed into a binary (0 and 1) function by applying a shift of $-a$ followed by a scaling of $b - a$. If the Fourier magnitude zero frequency term is not measured and the fill fraction constraint is appropriately applied, then the constraints

cannot distinguish an image from its shifted version. Since the zero frequency term is never available in experimental data only the scale factor is of concern.

In principle, the scale factor can be estimated simply by Fourier transforming the binary envelope and comparing the Fourier magnitudes with those measured. Clearly, if more Fourier magnitudes are compared the result will be more accurate. However, most of the information about the envelope is coded in the lower order Fourier magnitudes, and the total power in the higher order magnitudes are insensitive to the envelope shape.

Since protein molecules are globular, the following method was used to estimate the scale factor. The unit cell is partitioned into crystallographic asymmetric units, with the asymmetric unit chosen to minimize its aspect ratio, i.e. to be as close to cubic as possible. An ellipsoid is placed in each asymmetric unit, with the ratio of the semi-axes of the ellipsoid being equal to the ratio of the axes of the asymmetric unit, and the size of the ellipsoid is chosen such that the total volume of the ellipsoids is equal to the known volume of the envelope. For crystals of low solvent content the ellipsoids may be truncated by the faces of the asymmetric unit, and the size of the ellipsoid would then need to be increased. The scale factor is then estimated as

$$s = \sqrt{\frac{\sum\limits_{\mathbf{h}\in W'} |G[\mathbf{h}]|^2}{\sum\limits_{\mathbf{h}\in W'} M[\mathbf{h}]^2}}, \tag{4.6}$$

where the $|G[\mathbf{h}]|$ are the structure factor magnitudes of the ellipsoid model, $M[\mathbf{h}]$ denote the Fourier magnitudes of the envelope derived from the measured data, and W' is an appropriate resolution range.

The scaled Fourier magnitude projection P_B is then given by

$$P_B\mathbf{x} = \mathcal{F}^{-1}[\tilde{P}_B\mathcal{F}[\mathbf{x}]], \tag{4.7}$$

where $\tilde{P}_B$ is the Fourier magnitude projection in Fourier space given by

$$\tilde{P}_B X[\mathbf{h}] = \begin{cases} sM[\mathbf{h}]e^{i\angle X[\mathbf{h}]} & \text{if } \mathbf{h} \in W \\ X[\mathbf{h}] & \text{if } \mathbf{h} \notin W, \end{cases} \tag{4.8}$$

where $\angle\cdot$ denotes the phase, s is the scale factor, and W denotes the set of reciprocal lattice points where the data are measured (i.e. between the resolutions d_{min} and d_{max}).

4.3.3 Crystallographic symmetry

Using the Fourier and binary connectivity constraints, an image $x(t_1, t_2, t_3)$ and its shifted image $x(t_1 - d_1, t_1 - d_2, t_1 - d_3)$ cannot be distinguished. However, the shifted image may no longer satisfy the crystallographic symmetry. In this case, with $P2_12_12_1$ symmetry, only shift values which are multiples of half the unit cell length maintain the symmetry, i.e.

only $\mathbf{d} = (0.5k_1L_1, 0.5k_2L_2, 0.5k_3L_3)$, where L_1, L_2 and L_3 are the unit cell lengths and k are integers, will maintain the $P2_12_12_1$ crystallographic symmetry.

Another effect of the crystallographic symmetry is that it causes the Fourier data to be no longer independent, possibly resulting in additional uniqueness problems. In the case of $P2_12_12_1$ symmetry, the Fourier magnitudes at $|X(h_1, h_2, h_3)|$, $|X(-h_1, h_2, h_3)|$, $|X(h_1, -h_2, h_3)|$, $|X(h_1, h_2, -h_3)|$ are identical, i.e. there is a reflection symmetry of the Fourier magnitudes. Therefore the images $x(t_1, t_2, t_3)$, $x(-t_1, t_2, t_3)$, $x(t_1, -t_2, t_3)$ and $x(t_1, t_2, -t_3)$ cannot be distinguished by the constraints.

The crystallographic symmetry reduces the RMS power of the noise by the square root of the order of symmetry. If an image satisfies crystallographic symmetry, the binary, connectivity, and Fourier magnitude projections will also satisfy the crystallographic symmetry. If fN is a multiple of the order of the crystallographic symmetry (and therefore so is $(1-f)N$) then the fill fraction projection will also satisfy crystallographic symmetry. If not, the difference between the image after the fill fraction constraint has been applied and the image after both the fill fraction and crystallographic symmetry projections have been applied in that order is $\leq M/2$ pixels where M is the order of the symmetry, which is insignificant. The other possible source of error is rounding errors, mainly in the Fourier transform for the Fourier magnitude constraint. This should also be insignificant. The crystallographic symmetry constraint is therefore best enforced at the end of every iteration as described in Sec. 2.6.2, where it prevents the small errors from accumulating. Then it limits the search space for the algorithm.

4.4 Simulations

The algorithm described above was tested by simulation on two molecular envelopes derived from solved protein structures taken from the Protein Data Bank (PDB) [47]. The two proteins are the alkaline protease from *P. aeruginosa* [48] (PDB:1akl), and human Galectin-7 [49] (PDB:1bkz). For convenience, the two proteins are referred to here as proteins A and B, respectively. Both structures have space group $P2_12_12_1$ (the asymmetric unit is $1/4$ of the unit cell). The unit cell dimensions, sampling grid size, and grid spacings are listed in Table 4.1. The solvent excluded volumes for the two proteins are quite different, with $f = 0.35$ for protein A and $f = 0.57$ for protein B. Molecular envelopes were determined from the atomic models using standard procedures [50, 51], as implemented in the program "DM" [52], using an averaging radius of 8Å. The Fourier magnitudes of the envelope were calculated using the DFT, a scale factor applied, 5% RMS Gaussian noise added, and the magnitudes within a resolution shell between 40 and 7Å used as data for image reconstruction. The algorithm was started with a random binary image. Although the algorithm does not break any crystallographic symmetry present, the $P2_12_12_1$ crystallographic symmetry is still maintained by averaging the image over the 4 asymmetric units at the end of each iteration as described above. Protein B has an additional 2-fold non-crystallographic symmetry which can be seen as two compact areas in each asymmetric unit in Fig. 4.1(c),

Table 4.1 Parameters of the proteins A and B.

Protein	Cell Dimensions	Number of grid points	Grid Spacing	f
A	$77.2 \times 176.7 \times 51.1$Å	$18 \times 40 \times 12$	4.3 Å	0.35
B	$54.2 \times 65.3 \times 73.6$Å	$18 \times 20 \times 24$	3.1 Å	0.57

although this information was not used in the envelope reconstruction.

Since the estimate of the envelope $\hat{\mathbf{x}}_n$ (Eq. 2.28) satisfies the real space constraints, convergence of the algorithm was monitored by calculating the RMS Fourier magnitude error given by

$$R''_n = \sqrt{\frac{\sum\limits_{\mathbf{h}\in W} (|\hat{X}_n[\mathbf{h}]| - sM[\mathbf{h}])^2}{\sum\limits_{\mathbf{h}\in W} s^2 M[\mathbf{h}]^2}}, \tag{4.9}$$

where the $|\hat{X}_n[\mathbf{h}]|$ are the structure factor magnitudes of $\hat{\mathbf{x}}_n$. The error in the envelope given by

$$T_n = \frac{||\mathbf{x}_b - \hat{\mathbf{x}}_n||}{||\mathbf{x}_b||}, \tag{4.10}$$

where $\mathbf{x}_b$ is the true envelope, was also calculated to monitor the accuracy of the solution. The proportion of grid points in error is then equal to fT_n. Clearly, this metric can only be calculated for simulations where the original envelope is known. It was found that $T_n < 0.2$ corresponds to a good estimate of the true envelope. When calculating the image domain error metrics, the trivially-false solutions corresponding to inversion of the image in the origin, shifting of the image and the negative image $1 - \mathbf{x}$ were taken into account by applying the operations to the estimate and choosing the modified estimate with the lowest image error metric as described in Sec. 2.8.2.2. Since the Fourier magnitude error metrics provide a good measure of the progress of the algorithm, clearly showing convergence, the computationally expensive image domain error metrics were not calculated at each iteration but only at the end.

4.4.1 Scale factor determination

The scale factor was determined as described in Sec. 4.3.2. The true and ellipsoid envelopes used for proteins A and B are shown in Fig. 4.1. Plots of the the distribution of power in the Fourier magnitudes for the two proteins are shown in Fig. 4.2. The value on the y-axis is the total RMS energy in the Fourier magnitudes between $7\mathring{A}$ and the value on the x-axis. The energy distribution for two other models where the asymmetric unit is defined differently (i.e. does not minimize the aspect ratio) are also plotted.

It can be seen that the energies for all three models start to diverge when the lower limit of

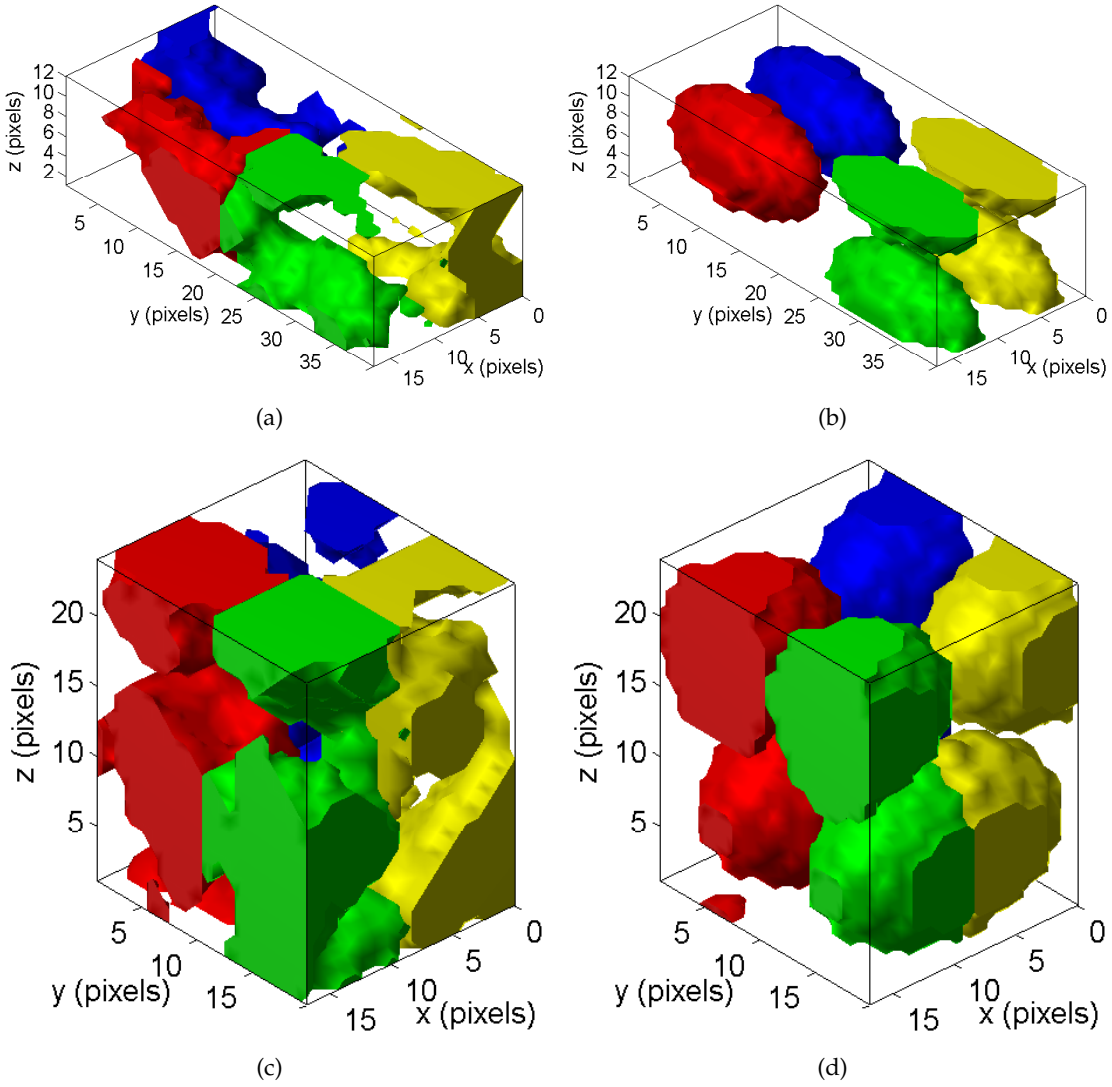


Figure 4.1 (a) Correct envelope for protein A, (b) Ellipsoid envelope for protein A, (c) Correct envelope for protein B, (d) Ellipsoid envelope for protein B.

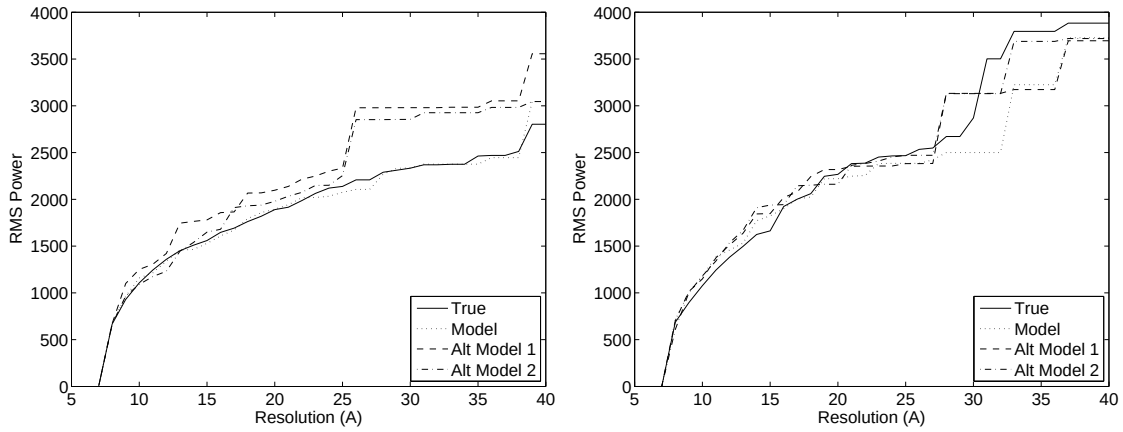


Figure 4.2 Total RMS energy in the Fourier magnitudes between 7Å and the value on the x-axis for (a) protein A (b) protein B. "Alt Model" 1 and 2 denote the two models where the asymmetric unit is not defined to minimize the aspect ratio.

the resolution falls to lower than 25$\mathring{A}$ for all envelopes, suggesting that the specific shape of the envelope is starting to affect the power distribution. A resolution range of 25 to 7Å was therefore used. Using Eq. 4.6, the scale factor was calculated to be 0.99 for protein A and 0.95 for protein B, the correct value being 1.0. With the added Gaussian noise, the energy in the Fourier magnitudes increases, which makes the estimated scale factor lower than the true value. With the 5% Gaussian noise, the scale factors were 0.94 and 0.97 for proteins A and B, respectively.

Starting with the estimated scale factors and refining them after each iteration sometimes led to divergence. Locking the scale factor gave better performance, and simulations showed that an error of 5% in the correct scale factor did not significantly affect convergence or the quality of the solution. The algorithm converged for both proteins when using their estimated scale factors of 0.94 and 0.99, but there was a minor improvement in the number of converged runs when the 0.94 scale factor for protein A was increased by 5%. A variety of scale factors can be tried if the algorithm does not converge with the estimated scale factor.

4.4.2 Effect of missing low resolution data

The fraction of the total energy in the Fourier magnitudes (not including the origin term) in some Fourier resolution ranges for the two proteins are shown in Table 4.2. It can be seen that a lot more energy resides in the lower resolution Fourier magnitudes than in the higher resolution Fourier magnitudes. This severely affects the performance of the algorithm although the presence of the connectivity constraint partially alleviates the problem as described in Sec. 3.4.4. There is very little energy in the low resolution Fourier magnitudes for protein B. The algorithm is therefore expected to converge faster than for protein

Table 4.2 Fraction of total energy of the Fourier magnitudes in each resolution range.

Protein	Resolution Range		
	Zero frequency to 50Å	Zero frequency to 40Å	7Å to 0Å
A	0.21	0.51	0.02
B	0	0.08	0.12

A, and this is later shown to be the case.

4.4.3 Noise

Because of the noise, there may be no solution which exactly satisfies all the constraints, and the nature of the DM algorithm is such that the iterates then move away from the near solution. Therefore a large number of iterations is used, and the solution with the minimum error metric R_n'' is chosen. Another approach would be to average over a few of the iterations after convergence and then apply the image domain constraints. In practice, it was found to be best to run the algorithm a few times with different starting envelopes and select the solution with the best agreement index R_n'' .

4.4.4 Negative solutions

As described in Sec. 3.3.3, if the solvent-excluded volume f is close to 0.5, the algorithm may converge to the negative solution (solvent and protein regions interchanged). Since convergence was rapid once in the vicinity of the solution, the method described in Sec. 3.3.3 was used successfully with $I_{short} = 50$ iterations and with a relatively conservative value of $I_{long} = 300$.

4.4.5 Results

The DM algorithm was run with $\beta = 0.9$ for 1.5×10^5 iterations with 5% noise on the data for envelopes A and B. Values $0.7 < |\beta| < 1.0$ worked well, however positive values of β gave slightly faster convergence than negative values. Five runs were made using different random starting envelopes for two sets of resolution ranges. The results are summarized in Table 4.3. The table shows the resolution range used, the total number of runs, the number of converged runs ($R_n'' < 0.25$), the number of successful converged runs ($T_n < 0.2$), and the number of incorrect solutions obtained.

For protein A, with data in the range 40 - 7Å, 2 of the 5 runs converged. For all of the converged runs, an accurate reconstruction of the envelopes was obtained with T_n around 0.03, i.e. no incorrect solutions were obtained. If the lower resolution limit is reduced to

Table 4.3 Results for the two protein envelopes.

Protein	Resolution range (Å)	Runs	Converged runs	Correct solutions	Incorrect solutions
A	40 - 7	5	2	2	0
A	50 - 7	5	5	5	0
B	40 - 7	5	5	1	4
B	50 - 7	5	5	5	0

Table 4.4 Results for the two protein envelopes. The bold type indicates a successful convergence.

Protein	Resolution range (Å)	Image error T_n	RMS Fourier Error R''_n
A	40 - 7	**0.033** 1.03 1.03 1.07 **0.028**	**0.24** 0.425 0.430 0.422 **0.230**
A	50 - 7	**0.034 0.041 0.045 0.034 0.032**	**0.184 0.197 0.212 0.191 0.189**
B	40 - 7	0.423 **0.140** 0.432 0.424 0.391	**0.181 0.182 0.181 0.182 0.182**
B	50 - 7	**0.069 0.067 0.086 0.086 0.090**	**0.135 0.139 0.138 0.140 0.132**

50Å, all runs converged to the correct solution. The results for one of the successful reconstructions for which $R''_n = 0.24$ and $T_n = 0.033$ are shown in Fig. 4.3. The plot of R''_n versus iteration shows a sharp drop at about iteration 25000, followed by erratic movement of the iterate around the correct solution. The true envelope and the reconstructed envelope are also shown in the figure. The reconstructed envelope is seen to be a good estimate of the true envelope. The algorithm has therefore been successful in this case.

For protein B, with data in the range 40 - 7Å, all 5 of the runs converged to low constraint-error values. Of these, 1 gave the correct solution ($T_n < 0.2$), and 4 gave incorrect solutions ($T_n > 0.2$). Therefore, although convergence could be obtained, the existence of multiple solutions that replicate the data indicates that, in this case, the data are insufficient to define a unique solution. All the false solutions are structurally similar as shown in Fig. 4.5, so the false results may be an artifact of the 2-fold non-crystallographic symmetry. Extending the lower resolution limit down to 50Å, all 5 runs converged, and all gave the correct solution ($T_n < 0.2$). In this case therefore, more low resolution diffraction data are needed to uniquely define the envelope. The results for one of the successful reconstructions for which $R''_n = 0.135$ and $T_n = 0.069$ are shown in Fig. 4.4. In this case the error R''_n drops quite rapidly and the algorithm is stable at the solution. The true envelope and the reconstructed envelope are also shown in the figure. Although $T_n = 0.069$, for the reconstructed envelope, the reconstruction is quite accurate with only 4% of the grid points misclassified. Convergence for protein B can be seen to be much faster than convergence for protein A as predicted in Sec. 4.4.2.

4.5 Conclusions

The structure factors of a molecular envelope obtained from solvent contrast variation experiments, when coupled with *a priori* information on envelopes, can uniquely define the molecular envelope. Incorporation of binary, fill fraction, connectivity and compactness constraints into an iterative projection algorithm gives an effective way of reconstructing envelopes from such data. Simulations with real envelopes and realistic levels of noise and missing data indicate that this algorithm may be practical. Advantages of the algorithm are that it is automatic and requires no additional information. The solution to the problem is sensitive to missing low resolution data and to an accurate determination of the scale factor.

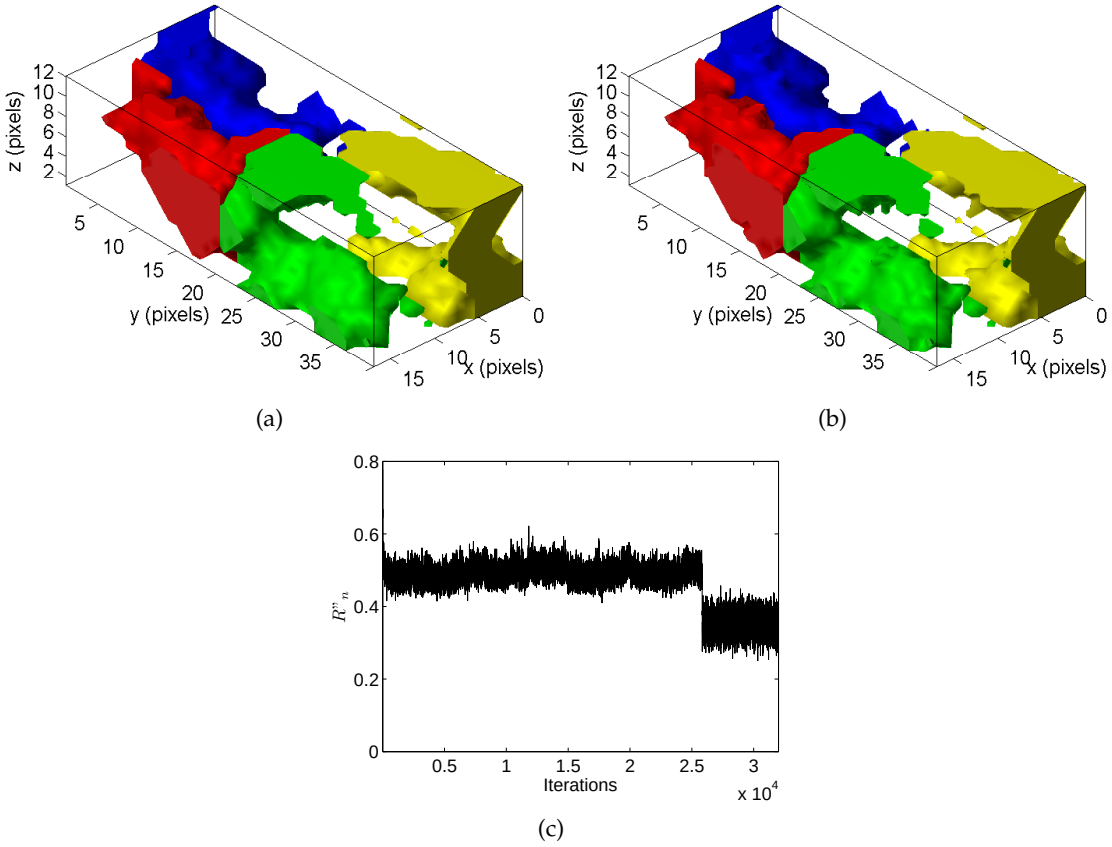


Figure 4.3 Original and reconstructed protein envelopes and R''_n vs iterations plot for protein A. At minimum R''_n, $T_n = 0.033$, $R''_n = 0.24$. (a) The true envelope, (b) the reconstructed envelope, and (c) R''_n versus iteration. Symmetry equivalent regions in the unit cell are represented by different colours to aid interpretation.

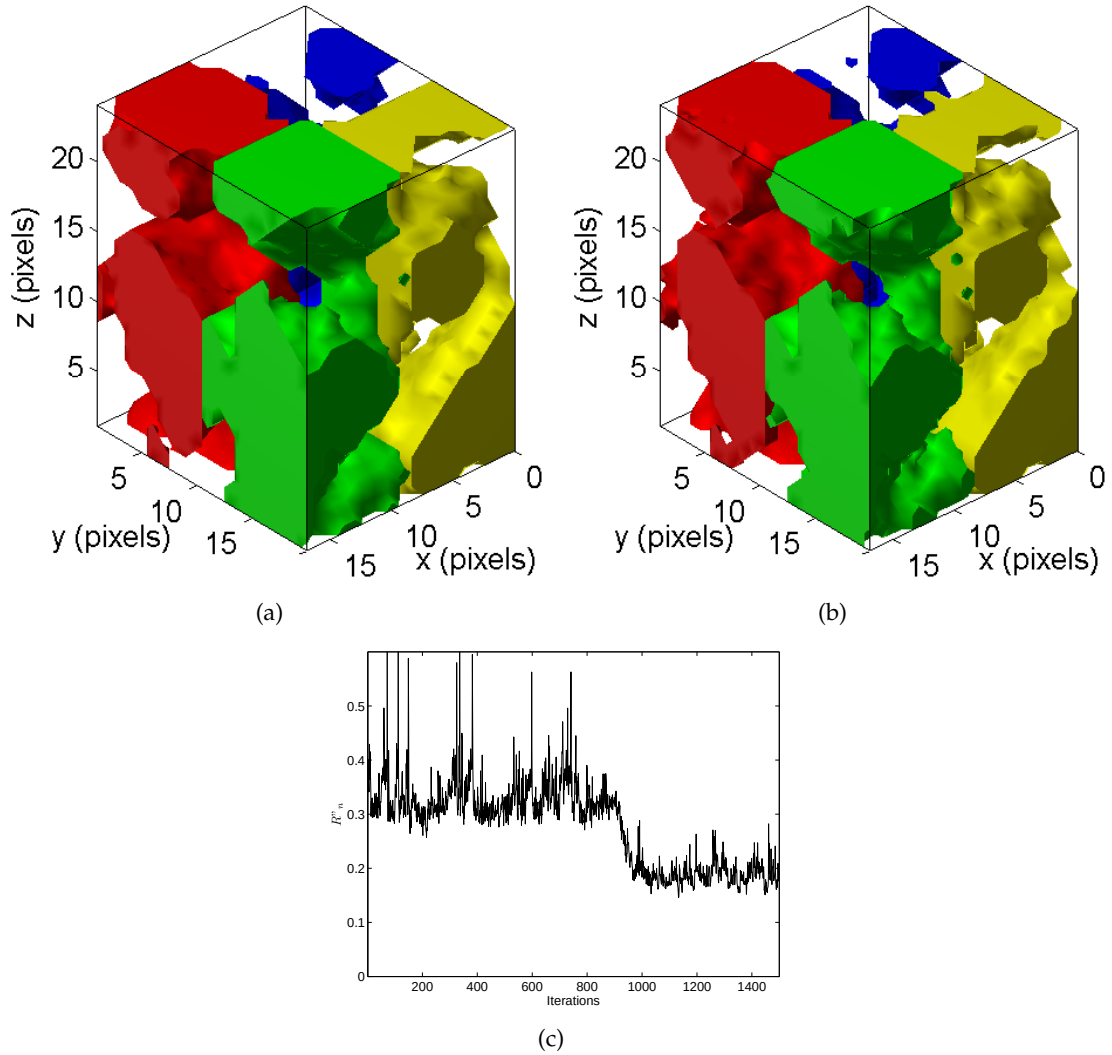


Figure 4.4 Original and reconstructed protein envelopes and R''_n vs iterations plot for protein B. At minimum R''_n, $T_n = 0.069$, $R''_n = 0.135$. (a) The true envelope, (b) the reconstructed envelope, and (c) R''_n versus iteration. Symmetry equivalent regions in the unit cell are represented by different colours to aid interpretation.

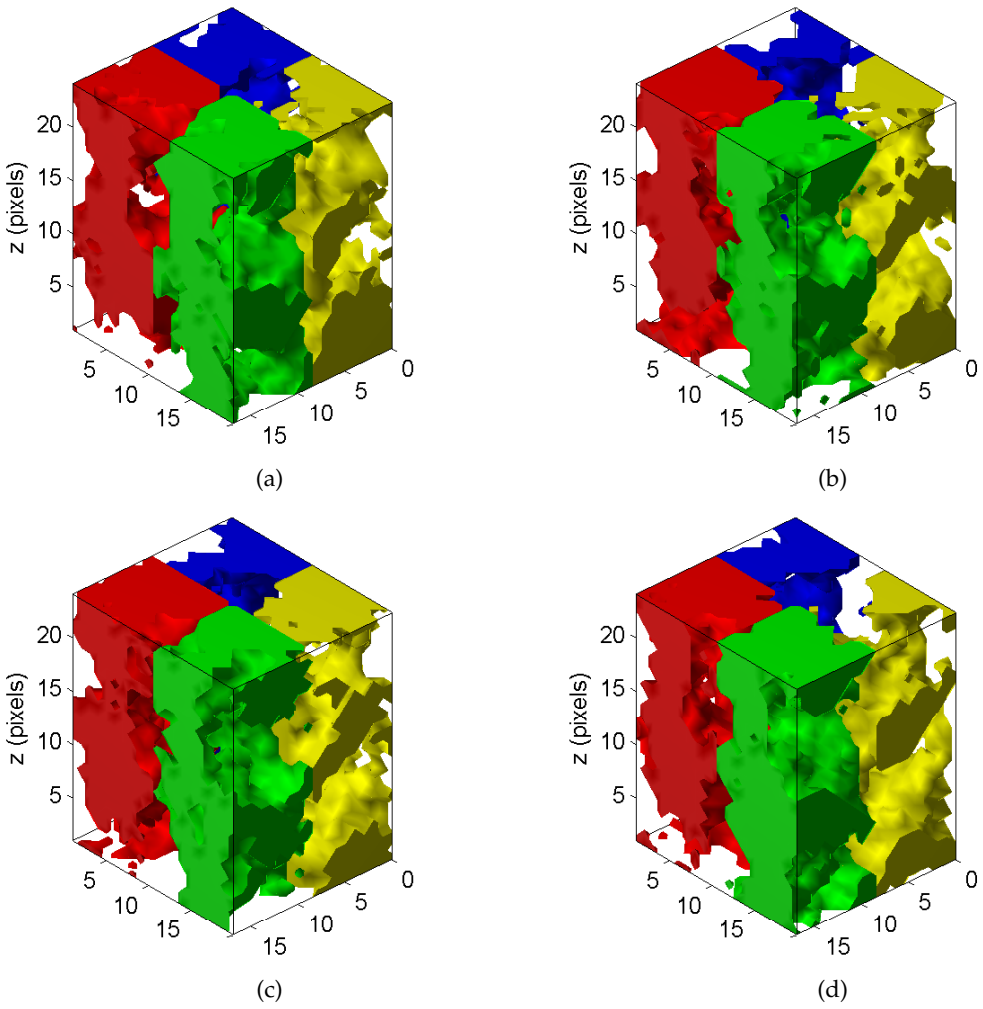


Figure 4.5 The four protein B envelopes which are not correct but which satisfy the Fourier magnitude data from 7-40Å.

Chapter 5

Symmetry as a Constraint in Image Reconstruction

5.1 Introduction

Extensive use is made of symmetry in structure determination in macromolecular x-ray crystallography [53]. Local symmetry properties of the molecule provide additional information that can compensate, in part, for the undersampling by the crystal diffraction amplitudes [54]. Local symmetry is used to refine the phase information derived by experimental methods (Sec. 1.2.4) and extend them to higher resolution. The algorithms used to do this are simple examples of projection algorithms [8]. In this chapter the effect of symmetry constraints on image reconstruction is studied in some detail.

The concepts and terminology of symmetry are first introduced. Symmetry projections and interpolation are then discussed. Finally, there is a discussion on possible methods for finding the symmetry parameters needed to enforce the symmetry.

5.2 Introduction to symmetry and symmetry constraints

An image **x** is called symmetric under a symmetry operation C if it is invariant under C, i.e.

$$\mathbf{x} = C\mathbf{x} \tag{5.1}$$

where C is the symmetry operator. Symmetry often arises if an image consists of two copies of an object whose positions are related by C. Eq. (5.1) is written in an equivalent form

$$x(\mathbf{t}) = x(C\mathbf{t}), \quad \forall \mathbf{t} \tag{5.2}$$

where $x(\mathbf{t})$ denotes the image **x** as a function of position **t**, and the two interpretations of the operator C are used interchangeably.

5.2.1 Symmetry operations

A symmetry operation on an image is a rigid body, or distance preserving, transformation consisting of a combination of *reflection*, *rotation*, and *shift* operations applied to the entire image. It can be shown that at most one of each operation is needed for any rigid body operation. The rigid body symmetry operation maps each point in the image to its symmetry-related point, and can be written as

$$C\mathbf{t} = \mathbf{R}(\mathbf{t} + \mathbf{R}^{-1}\mathbf{d}) \tag{5.3}$$

$$= \mathbf{R}\mathbf{t} + \mathbf{d}, \tag{5.4}$$

where $\mathbf{R}$ is a rotation (about the origin) and reflection matrix and $\mathbf{d}$ is a shift vector. The two equations show the shift applied before and after the rotation and reflection operation. If the centre of the rotation is at $\mathbf{D} = (\mathbf{I} - \mathbf{R})^{-1}\mathbf{d}$, then

$$C\mathbf{t} = \mathbf{R}(\mathbf{t} - \mathbf{D}) + \mathbf{D} \tag{5.5}$$

$$= \mathbf{R}\mathbf{t} + (\mathbf{I} - \mathbf{R})\mathbf{D}, \tag{5.6}$$

where $\mathbf{I}$ is the identity matrix. A symmetry is *proper* if the symmetry operations can be carried out without using a reflection, i.e. chirality is preserved.

5.2.2 Symmetry groups

Since the image is invariant under the symmetry operation, then it must also be invariant under repeated applications of the symmetry operation. The set of all repeated symmetry operations forms a *symmetry group*, $C = \{C_1, C_2, ..., C_q\}$. Choosing one of the symmetry operations in the symmetry group C and denoting it C_1, repeated applications of C_1 can be used to generate the group, i.e.

$$C_k = C_1^k \tag{5.7}$$

where C_1^k denotes k applications of the C_1 operator. If C is a *complete* symmetry, then after some minimum number q of symmetry operations the cycle repeats itself irrespective of which element in the set was chosen as the C_1 symmetry operation, i.e.

$$C_q = C_1^q = I \tag{5.8}$$

where I is the identity operator. C therefore forms a finite group of order q given by $C = \{C_1, C_2, ..., C_q\}$. Note that if q has two factors $q = q_1 q_2$, then the symmetry operations $C_k^{q_2}$ form a group of order q_1, and vice versa. If C is an *incomplete* symmetry, then the cycle does not repeat itself, and the group has infinite order. Only finite groups are considered here. A group may have a number of C_1's, e.g. 5-fold, 3-fold, and 2-fold in an icosahedral symmetry, for example. The set of symmetry operations involving rotations about a single point is referred to as a *point group*, so the centre of rotation $\mathbf{D}$ in Eq. (5.5) is the same for all members in the point group.

(a) (b) (c)

Figure 5.1 Symmetries (a) Complete global symmetry (b) Complete local symmetry which exists over the grey area (c) Incomplete local symmetry which exists over the grey area

5.2.3 Global and local symmetry

It is necessary to distinguish between *global* and *local* symmetries. A symmetry is global if it applies over an infinite domain. Thus if **x** is an image which satisfies the global symmetry C, then

$$x(\mathbf{t}) = x(C\mathbf{t}) \quad \forall\, \mathbf{t}. \tag{5.9}$$

An example of global symmetry is shown in Fig. 5.1(a).

A local symmetry applies only over a restricted domain Q of the image, i.e.

$$x(\mathbf{t}) = x(C\mathbf{t}) \quad \mathbf{t} \in Q, \tag{5.10}$$

where Q denotes the subset of the image domain over which the symmetry applies. An example of complete local symmetry is shown in Fig. 5.1(b).

A global symmetry must be complete. If this were not the case then repeated applications of the symmetry would result in a symmetry set which only a uniform image would satisfy. On the other hand, a local symmetry may not necessarily be complete as shown in Fig 5.1(c), where a symmetry operator consisting of a 120° rotation applies in a subset which consists of 2/3 of a circle. Combinations of both global and local symmetries are possible.

5.2.4 Crystallographic and non-crystallographic symmetry

A crystallographic symmetry is a set of global symmetry operators that are consistent with translational symmetry (periodicity). This requirement is quite restrictive and there are only 17 crystallographic symmetries in 2D and 230 crystallographic symmetries in 3D [7]. These are referred to as crystallographic *plane groups* and *space groups*, respectively. For a crystallographic symmetry, it is possible to sample the image with a regular lattice such

that the symmetry related point of every sample point is also a sample point.

Local symmetries that apply only over a part of a periodic image are often called non-crystallographic symmetries (NCS). A periodic image may contain both crystallographic and non-crystallographic symmetries.

As described in Sec. 1.2.3, a crystallographic symmetry of order q creates redundancy in the Fourier magnitudes of order q, and so no additional information (phasing power) is gained. On the other hand, the symmetry equation for a local symmetry over a restricted domain Q requires multiplication of the image domain object by the mask Q, i.e.

$$\begin{aligned} \mathbf{x}Q &= C(\mathbf{x}Q) && (5.11)\\ \mathbf{x} &= C(\mathbf{x}Q) + \mathbf{x}(1-Q). && (5.12) \end{aligned}$$

The image domain multiplication is a convolution in the Fourier domain, i.e.

$$\begin{aligned} \mathbf{X} \otimes Q &= C(\mathbf{X} \otimes Q) && (5.13)\\ \mathbf{X} &= C(\mathbf{X} \otimes Q) + \mathbf{X} \otimes (1-Q). && (5.14) \end{aligned}$$

The convolution means that the phases of the Fourier samples $\mathbf{X}$ affect the magnitudes of the other Fourier samples. In other words, information about the *phases* at each Fourier sample is encoded in the other Fourier *magnitudes*. So local symmetry, or NCS, provides additional information, or phasing power.

5.2.5 Symmetry on continuous and sampled images

If the image $x(\mathbf{t})$ exists on a continuous domain $\mathbf{t}$ then Eq. (5.2) is well defined. Since for practical computation the image $x(\mathbf{t})$ must be sampled, $C\mathbf{t}$ may not be a sample point making Eq. (5.2) poorly defined. One approach to this difficulty is to define $x[C\mathbf{t}]$ in terms of $x(\mathbf{t})$ at grid points $\mathbf{t}$ by interpolation. Generally, no interpolation is needed for crystallographic symmetry if the sampling is chosen appropriately. Interpolation is generally needed for NCS.

Due to the nature of the interpolation operation, it is difficult to consider the constraint set without considering the symmetry projection operation as well. Examples of applying the interpolated projection are now considered, and the constraint sets discussed alongside.

5.3 Symmetry projections

The projection $P_{Sym}\mathbf{x} = \mathbf{x}'$ onto the set of continuous images satisfying a symmetry C is

$$x'(\mathbf{t}) = \frac{1}{|C(\mathbf{t})|} \sum_{\mathbf{u} \in C(\mathbf{t})} x(\mathbf{u}), \qquad (5.15)$$

where $C(\mathbf{t})$ is the set of points symmetry related to $\mathbf{t}$. Since $q = |C(\mathbf{t})|$ is constant for the symmetries found in crystallography, Eq. (5.15) can be written for sampled crystallographic electron densities as

$$x'[\mathbf{t}] = \begin{cases} \frac{1}{q}\sum_{k=1}^{q} x[C_k\mathbf{t}] & \text{for } \mathbf{t} \in Q \\ x[\mathbf{t}] & \text{for } \mathbf{t} \notin Q \end{cases} \tag{5.16}$$

where Q is the set of points over which the symmetry applies. As described previously, if the image is sampled then $C_k\mathbf{t}$ may not be a sample point, so interpolation may be required to calculate $x[C_k\mathbf{t}]$ in Eq. (5.16).

5.3.1 Example projection for a 2D sampled image

Consider a q-fold rotation about the origin for a 2-D non-periodic sampled image where the symmetry extends over the whole of the image. Assume that the image is sampled uniformly with a unitary Cartesian sample spacing. Then

$$\begin{aligned} C_k\mathbf{t} &= R_k\mathbf{t} \quad \text{for } k = 1, 2, ..., q \\ &= \begin{pmatrix} \cos(2\pi k/q) & -\sin(2\pi k/q) \\ \sin(2\pi k/q) & \cos(2\pi k/q) \end{pmatrix} \mathbf{t} \\ &= \begin{pmatrix} c_q^k & -s_q^k \\ s_q^k & c_q^k \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \end{aligned} \tag{5.17}$$

. (5.18)

where $c_q^k = \cos(2\pi k/q)$ and $s_q^k = \sin(2\pi k/q)$. An element in the projected image is given by

$$\begin{aligned} x'[t_1, t_2] &= \frac{1}{q}\sum_{k=1}^{q} x[C_k\mathbf{t}] \\ &= \frac{1}{q}\sum_{k=1}^{q} x[R_k\mathbf{t}] \\ &= \frac{1}{q}\sum_{k=1}^{q} x[t_1c_q^k - t_2s_q^k, t_1s_q^k + t_2c_q^k] \end{aligned} \tag{5.19}$$

Unless $q = 2$ or 4, the arguments in Eq. (5.19) are not integers, and so interpolation is needed to calculate the image. Since the cases $n = 2, 4$ are trivial, it is assumed in the following that the symmetry operators do not map sample points to sample points.

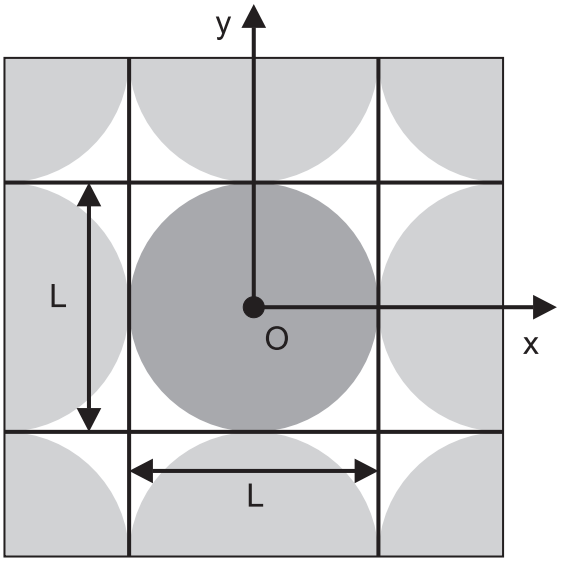


Figure 5.2 A periodic 2D image with the unit cell of side length L centred at the origin. The area Q over which the NCS applies is shown in grey.

5.3.2 Example projection for a 2D sampled periodic (or finite extent) image

Consider a q-fold rotation about the origin for a 2-D periodic sampled image. The symmetry only applies in the incircle of the square periodic unit, and the unit cell has side length L and is centred at the origin as shown in Fig. 5.2. The rotation matrix is given by

$$\begin{aligned} C_k\mathbf{t} &= R_k\mathbf{t} \quad \text{for } k = 1, ..., q \\ &= \begin{pmatrix} \cos(2\pi k/q) & -\sin(2\pi k/q) \\ \sin(2\pi k/q) & \cos(2\pi k/q) \end{pmatrix} \mathbf{t} \\ &= \begin{pmatrix} c_q^k & -s_q^k \\ s_q^k & c_q^k \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}. \end{aligned} \tag{5.20}$$

Then each element in the central period (unit cell) of the projected image is given by

$$\begin{aligned} x'[t_1, t_2] &= \begin{cases} \frac{1}{q}\sum_{k=1}^{q} x[C_k\mathbf{t}] & \text{if } \sqrt{t_1^2 + t_2^2} < L/2 \\ x[t_1, t_2] & \text{otherwise} \end{cases} \\ &= \begin{cases} \frac{1}{q}\sum_{k=1}^{q} x[t_1 c_q^k - t_2 s_q^k, t_1 s_q^k + t_2 c_q^k] & \text{if } \sqrt{t_1^2 + t_2^2} < L/2 \\ x[t_1, t_2] & \text{otherwise,} \end{cases} \end{aligned} \tag{5.21}$$

with those in the other unit cell being given by the appropriate translation. Again, unless

$q = 2$ or 4, the arguments in Eq. (5.21) are not integers, and so interpolation is needed to calculate the image.

5.3.3 Sinc interpolation for a sampled, infinite extent image

Assume that the image $\mathbf{x}$ is a sampled version of an underlying continuous image $\check{\mathbf{x}}$ which is circularly bandlimited with unit bandwidth, i.e.

$$\check{X}(h_1, h_2) \quad , \quad \text{for } \sqrt{h_1^2 + h_2^2} > 0.5, \tag{5.22}$$

where $\check{X}(h_1, h_2)$ is the value at $\mathbf{h} = (h_1, h_2)$ of the continuous Fourier transform $\check{\mathbf{X}} = \mathcal{F}\{\check{\mathbf{x}}\}$. The Nyquist criterion is satisfied (since the image sample spacing is unity) and sinc interpolation perfectly reconstructs the image. Eq.(5.21) is then

$$x'[t_1, t_2] = \frac{1}{q} \sum_{k=1}^{q} \sum_{p_1, p_2 = -\infty}^{\infty} x[p_1, p_2] \operatorname{sinc}(t_1 c_q^k - t_2 s_q^k - p_1) \operatorname{sinc}(t_1 s_q^k + t_2 c_q^k - p_2) \tag{5.23}$$

Note that the sum of the interpolation coefficients for each pixel is unity, i.e.

$$\sum_{p_1, p_2 = -\infty}^{\infty} \operatorname{sinc}(t_1 c_q^k - t_2 s_q^k - p_1) \operatorname{sinc}(t_1 s_q^k + t_2 c_q^k - p_2) = 1. \tag{5.24}$$

Since the Nyquist sampling theorem is satisfied as described in Eq.(5.22), it is possible to perfectly reconstruct the continuous image $\check{\mathbf{x}}$ from the sampled image $\mathbf{x}$. Furthermore, the rotated continuous images $R_k \check{\mathbf{x}}$ are also bandlimited, i.e. satisfy Eq. (5.22). Let ψ_k be the result of sampling the rotated continuous image, i.e.

$$\begin{aligned}
\psi_k[t_1, t_2] &= \check{x}(R_k \mathbf{t}) \quad \text{for } t_1, t_2 \in \mathbf{Z} && (5.25)\\
&= \sum_{p_1, p_2 = 0}^{\infty} x[p_1, p_2] \operatorname{sinc}(t_1 c_q^k - t_2 s_q^k - p_1) \operatorname{sinc}(t_1 s_q^k + t_2 c_q^k - p_2) && (5.26)\\
&= x(R_k \mathbf{t}). && (5.27)
\end{aligned}$$

Then the sinc-interpolated projection can be written

$$\begin{aligned}
x'[t_1, t_2] &= \frac{1}{q} \sum_{k=1}^{q} x[R_k \mathbf{t}] \quad \text{where } x[R_k \mathbf{t}] \text{ is found by sinc interpolation} && (5.28)\\
&= \frac{1}{q} \sum_{k=1}^{q} \psi_k[\mathbf{t}] && (5.29)\\
&= \frac{1}{q} \sum_{k=1}^{q} \check{x}(R_k \mathbf{t}) \quad \text{for } t_1, t_2 \in \mathbf{Z} && (5.30)\\
&= \check{x}'(R_k \mathbf{t}) \quad \text{for } t_1, t_2 \in \mathbf{Z}. && (5.31)
\end{aligned}$$

So $x'[t_1, t_2]$ is a sampled version of an underlying image $\check{\mathbf{x}}'$ that satisfies the symmetry, i.e. $\check{\mathbf{x}}'(R_k\mathbf{t}) = \check{\mathbf{x}}'(\mathbf{t})\ \forall k$. Hence the projected (with sinc interpolation) sampled image is identical to the sampled projected continuous image. This is useful because the symmetry is rigorously defined for a continuous image. Therefore a second application of the projection to $\mathbf{x}'$ leaves it unchanged. So the constraint set for sinc-interpolated symmetry can be rigorously defined as the set of (band-limited) images $\mathbf{x}$ that are invariant when the sinc interpolated symmetry projection is applied, i.e. $\mathbf{x}' = \mathbf{x}$, and Eq. (5.23) is the rigorous projection onto this constraint.

5.3.4 Sinc interpolation for sampled finite extent or periodic images

If $\mathbf{x}$ is of finite extent, or a periodic image, also bandlimited, with unit cell length $\mathbf{L} = (L_1, L_2)$ and $L_1 \times L_2 = N$ pixels, then the symmetry applies only over a region Q of the unit cell. Then $x'(\mathbf{t}) = x(\mathbf{t})$ for $\mathbf{t} \notin Q$ as described in Eq. (5.21), and perfect sinc interpolation can still be applied for $\mathbf{t} \in Q$ with

$$\begin{aligned} x'[t_1, t_2] &= \frac{1}{q}\sum_{k=1}^{q}\sum_{p_1=-\infty}^{\infty}\sum_{p_2=-\infty}^{\infty} x\left[L_1\left\{\frac{t_1c_q^k - t_2s_q^k}{L_1}\right\}, L_2\left\{\frac{t_1s_q^k + t_2c_q^k}{L_2}\right\}\right] \qquad (5.32)\\ &\quad \operatorname{sinc}(t_1c_q^k - t_2s_q^k - p_1)\operatorname{sinc}(t_1s_q^k + t_2c_q^k - p_2)\\ &= \sum_{p_1=0}^{L_1-1}\sum_{p_2=0}^{L_2-1} A(t_1, t_2, p_1, p_2)x[p_1, p_2], \qquad (5.33) \end{aligned}$$

where $\{\cdot\}$ is the fractional part operation, and the interpolation coefficients A are given by

$$\begin{aligned} A(t_1, t_2, p_1, p_2) = \ &\tfrac{1}{q}\sum_{k=1}^{q}\sum_{r_1,r_2=-\infty}^{\infty} \operatorname{sinc}(t_1c_q^k - t_2s_q^k - (p_1 + r_1L_1))\\ &\operatorname{sinc}(t_1s_q^k + t_2c_q^k - (p_2 + r_2L_2)) \quad \forall r_1, r_2 \in \mathbb{Z}. \qquad (5.34) \end{aligned}$$

Note that $\sum_{p_1,p_2=0}^{\mathbf{L}-1} A(t_1, t_2, p_1, p_2) = 1$. By precalculating the A values as shown, it is possible to avoid doing an infinite sum at each iteration, but sinc interpolation still requires summation over every pixel in the image to find the value at each point. This is impractical for projection algorithms which require many iterations.

It can be shown, similarly to Sec. 5.3.3, that the projected image $x'[t_1, t_2]$ is identical to the samples of a symmetry projected underlying continuous image, so that in this case also the symmetry constraint is rigorously defined and the projection is rigorous and unique.

5.3.5 Linear interpolation

In most cases sinc interpolation is too computationally expensive. Linear interpolation is commonly used because of its computational simplicity and because it is reasonably

accurate for small sample spacings.

In the 2D case, symmetry averaging by nearest neighbour bilinear interpolation gives

$$\begin{aligned} x'[t_1,t_2] &= \frac{1}{q}\sum_{k=1}^{q} x\left[\lfloor t_1c_q^k - t_2s_q^k\rfloor, \lfloor t_1s_q^k + t_2c_q^k\rfloor\right] \langle t_1c_q^k - t_2s_q^k\rangle\langle t_1s_q^k + t_2c_q^k\rangle \\ &+ x\left[\lfloor t_1c_q^k - t_2s_q^k\rfloor, \lceil t_1s_q^k + t_2c_q^k k\rceil\right] \langle t_1c_q^k - t_2s_q^k\rangle(1 - \langle t_1s_q^k + t_2c_q^k\rangle) \\ &+ x\left[\lceil t_1c_q^k - t_2s_q^k\rceil, \lfloor t_1s_q^k + t_2c_q^k\rfloor\right] (1 - \langle t_1c_q^k - t_2s_q^k\rangle)\langle t_1s_q^k + t_2c_q^k\rangle \\ &+ x\left[\lceil t_1c_q^k - t_2s_q^k\rceil, \lceil t_1s_q^k + t_2c_q^k\rceil\right] (1 - \langle t_1c_q^k - t_2s_q^k\rangle)(1 - \langle t_1s_q^k + t_2c_q^k\rangle) \quad (5.35) \\ &= \frac{1}{q}\sum_{k=1}^{q} x[R_k^L\mathbf{t}]. \quad (5.36) \end{aligned}$$

where $\lceil\cdot\rceil$ denotes rounding up to the nearest integer, $\lfloor\cdot\rfloor$ denotes rounding down, $\langle\cdot\rangle$ denotes the fractional part, and $R_k^L\mathbf{t}$ means linear interpolation to obtain $R_k\mathbf{t}$.

The symmetry projection using linear interpolation for a sampled, finite extent (or periodic) image is given by

$$x'[t_1,t_2] = \begin{cases} \frac{1}{q}\sum_{k=1}^{q} x[R_k^L\mathbf{t}], & \text{for } \mathbf{t}\in Q \\ x[t_1,t_2], & \text{for } \mathbf{t}\notin Q, \end{cases} \quad (5.37)$$

where Q is the region of the unit cell over which the symmetry applies.

Consider now the linear interpolation projection operation for a sampled, infinite extent image where the symmetry applies over the entire image. Define a new image ψ_k where the value at each grid point in $\psi_k[\mathbf{t}]$ is the interpolated value of $x[R_k^L\mathbf{t}]$, i.e.

$$\psi_k[t_1,t_2] = x(R_k^L\mathbf{t}), \quad (5.38)$$

so

$$x'[t_1,t_2] = \frac{1}{q}\sum_{k=1}^{q}\psi_k[t_1,t_2]. \quad (5.39)$$

Let $\acute{\mathbf{x}}$ denote the continuous analog (i.e. written as a function of a continuous variable) of a sampled image $\mathbf{x}$, i.e.

$$\acute{x}(t_1,t_2) = \sum_{p_1,p_2=-\infty}^{\infty} x[p_1,p_2]\delta(t_1-p_1,t_2-p_2), \quad (5.40)$$

where p_1 and p_2 are integers. Then $\acute{\mathbf{x}} = \check{\mathbf{x}}\kappa$, where κ is the comb function as defined in Eq. (1.6).

Then ψ_k can be written as sampling a continuous convolution between $R_k\acute{\mathbf{x}} = \acute{x}(R_k\mathbf{t})$ and

a tri function,

$$\psi_k \quad = \quad (R_k \acute{\mathbf{x}} \otimes \text{tri})\kappa \tag{5.41}$$

or

$$\psi_k[t_1, t_2] \quad = \quad \acute{x}(R_k \mathbf{t}) \otimes \text{tri}(\mathbf{t}) \quad \text{for } t_1, t_2 \in \mathbb{Z} \tag{5.42}$$

$$= \iint_{-\infty}^{\infty} \acute{x}(z_1 c_q^k - z_2 s_q^k, z_1 s_q^k + z_2 c_q^k)\ \text{tri}(t_1 - z_1, t_2 - z_2)\ dz_1 dz_2, \text{ for } t_1, t_2 \in \mathbb{Z} \tag{5.43}$$

where the tri function is defined as

$$\text{tri}(\mathbf{t}) = \begin{cases} (1 - |t_1|)(1 - |t_2|) & \text{for } -1 \leq t_1, t_2 \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.44}$$

The result is that when the symmetry averaging is applied, the convolution with the tri function brings weighted averages at grid points adjacent to symmetry-related non-grid points into the average. If the symmetry averaging is repeated, the same occurs and the result continues to change. The operation is therefore not idempotent. In fact, the repeated averaging at each application of the projection leads to a constant image in the limit. Furthermore, since linear interpolation does not correspond to "perfect" interpolation of any underlying continuous image, there is no underlying rigorous constraint set.

The above analysis can be conducted more rigorously in the Fourier domain as follows. The Fourier transform of κ is also κ, and the Fourier transform of a tri function is a sinc-squared function given by

$$\text{sinc}^2(h_1, h_2) = \left(\frac{\sin(\pi h_1)}{\pi h_1}\right)^2 \left(\frac{\sin(\pi h_2)}{\pi h_2}\right)^2, \tag{5.45}$$

so Eq. (5.41) can be written in the Fourier domain as

$$\begin{aligned} \Psi_k &= \mathcal{F}\{\psi_k\} \\ &= \mathcal{F}\{(R_k^L \acute{\mathbf{x}} \otimes \text{tri})\kappa\} \\ &= \mathcal{F}\{R_k^L \acute{\mathbf{x}}\}\mathcal{F}\{tri\} \otimes \mathcal{F}\{\kappa\} \qquad (5.46) \\ &= R_k^L \acute{\mathbf{X}}\ \text{sinc}^2 \otimes \kappa, \qquad (5.47) \end{aligned}$$

where $\acute{\mathbf{X}}$ is the (periodic) Fourier transform of $\acute{\mathbf{x}}$. Then the Fourier transform $\mathbf{X}' = \mathcal{F}\{\mathbf{x}'\}$ of the projection $\mathbf{x}'$ is given by

$$X'(h_1, h_2) \quad = \quad \frac{1}{q} \sum_{k=1}^{q} \Psi_k(h_1, h_2) \quad \text{for } h_1, h_2 \in \mathbb{Z} \tag{5.48}$$

$$X'[\mathbf{h}] \quad = \quad \frac{1}{q} \sum_{k=1}^{q} X[R_k^L \mathbf{h}]\text{sinc}^2(\mathbf{h}) \tag{5.49}$$

The multiplication of **X** by the sinc-squared function is a low pass filter and smooths the features in the image. V repeated applications of the projection is equivalent to multiplying the Fourier magnitudes by a $\text{sinc}^{2V}[\mathbf{h}]$ low pass filter. Since $\text{sinc}^{2V}[\mathbf{h}] \to \delta[\mathbf{h}]$ as $V \to \infty$, the image approaches a constant as the linear interpolated symmetry projection is repeated.

For sampled images in a finite extent, a similar analysis can be done, with the similar result that the pixels in the symmetry region Q tend to a constant when the linear interpolated projection is applied many times.

An example of repeated applications of a 3-fold NCS projection with linear interpolation to a finite extent image is shown in Fig. 5.3. The low pass filtering effect is easily seen, and the image becomes uniform in Q after around 5000 iterations.

5.4 Convexity of the symmetry constraint

The sampled non-interpolated symmetry constraint forces sets of q pixels to be equal, and is therefore a convex constraint of infinite extent. The problem is more difficult for interpolated symmetry. If sinc interpolation is used, then by making use of the idempotent property of projections, an image which satisfies an interpolated symmetry constraint is defined as an image which is invariant when a particular symmetry projection is applied to it. The interpolated symmetry constraint set is then the intersection of a set of linear equations, one for each pixel in the image, i.e. the sinc interpolated symmetry constraint set B is defined by

$$B = \{\mathbf{x} : \mathbf{x} = P_{Sym}\mathbf{x}\} \tag{5.50}$$

$$B = \{\mathbf{x} : x(\mathbf{t}) = \sum_{\mathbf{p}} A(\mathbf{t}, \mathbf{p})x(\mathbf{p}), \quad \forall\, \mathbf{t}\} \tag{5.51}$$

where $A(\mathbf{t}, \mathbf{p})$ is given by Eq. (5.34). Each linear equation is a convex constraint of infinite extent, and therefore so is their intersection, making B a convex set.

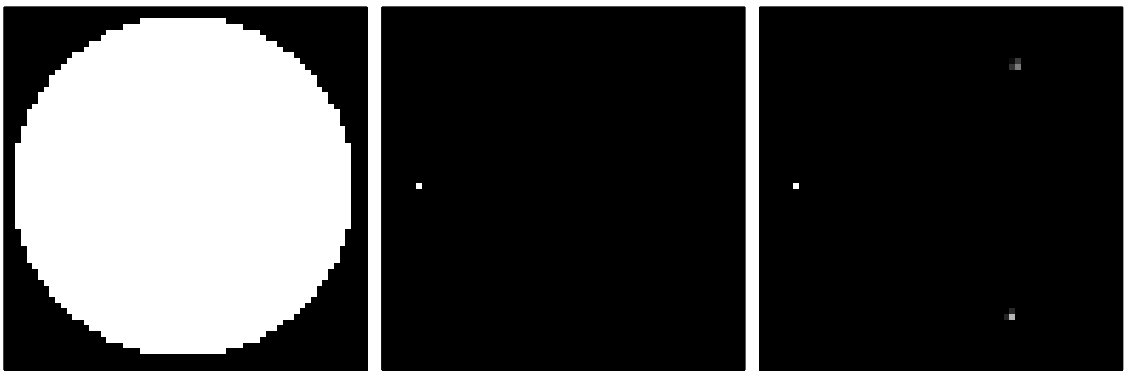
If linear interpolation is used, then the projection operation is no longer idempotent. However, a constraint set $B(\epsilon)$ can be defined as

$$B(\epsilon) = \{\mathbf{x} : |x[\mathbf{t}] - x'[\mathbf{t}]| < \epsilon, \quad \forall\, \mathbf{t}\}, \tag{5.52}$$

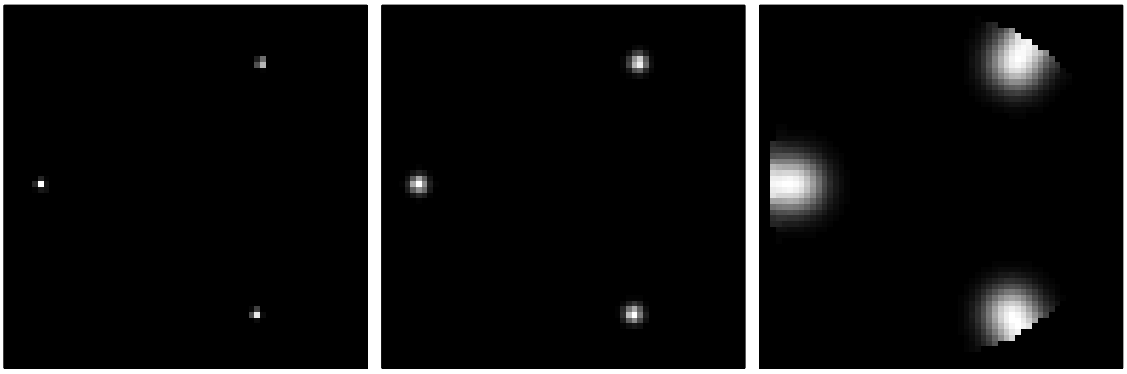
where ϵ is a tolerance and $x'[\mathbf{t}]$ is as defined in Eq. (5.37), i.e. the result of a symmetry projection using linear interpolation. Alternatively, Eq. (5.52) can be written as

$$B(\epsilon) = \{\mathbf{x} : ||P^L_{sym}\mathbf{x} - \mathbf{x}||_\infty < \epsilon\} \tag{5.53}$$

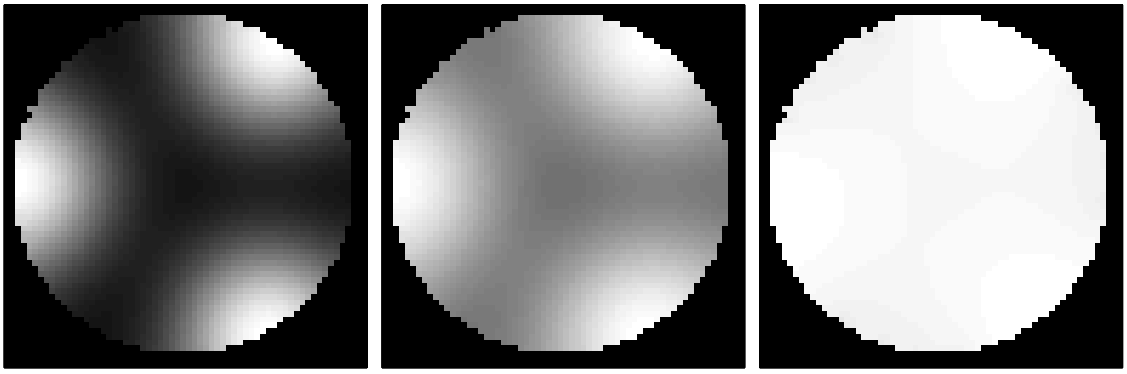
where P^L_{sym} denotes the symmetry projection using linear interpolation and $||\cdot||_\infty$ denotes the infinity norm or the largest value operator. The constraint set is then the intersection of a set of linear inequalities. Since each linear inequality is a convex constraint, their inter-



(a) NCS Support (b) Initial iterate (c) After 1 iteration



(d) After 2 iterations (e) After 10 iterations (f) After 100 iterations



(g) After 1000 iterations (h) After 2000 iterations (i) After 5000 iterations

Figure 5.3 Repeated applications of the interpolated NCS projection leads to a uniform image.

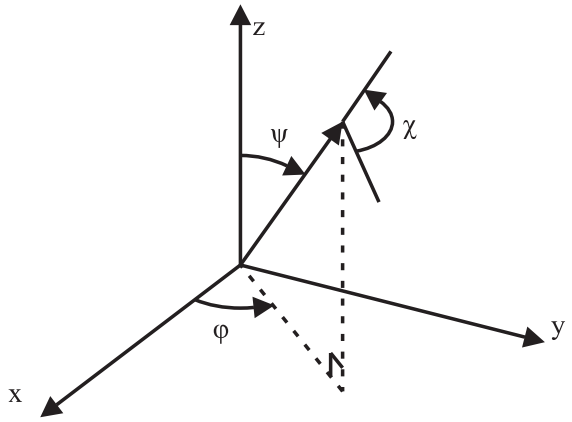


Figure 5.4 Rotation Axes

section $B(\epsilon)$ is also a convex set. With an appropriately chosen ϵ, the projection operator P_{sym}^{L} takes **x** to a point in $B(\epsilon)$.

In conclusion the symmetry constraints are either convex constraints of infinite extent under the definition of symmetry for an underlying bandlimited continuous image for sinc interpolation, or under the definition $A(\epsilon)$ for linear interpolation.

5.5 Rotation in 3-D space

In the remainder of this chapter, and in Ch. 6, rotation operators in 3D space are used. For convenience, this operator is defined here. A rotation about the origin in 3-D space can be written as a rotation of angle χ about a unit vector **u** as shown in Fig. 5.4. The coordinates of the unit rotation vector are given by $\mathbf{u} = (u_1, u_2, u_3) = (\cos\phi \sin\psi, \sin\phi \sin\psi, \cos\psi)$, and the rotation matrix R is then

$$R(\phi, \psi, \chi) = \begin{pmatrix} 1 + (u_1^2 - 1)(1 - \cos\chi) & -u_3 \sin\chi + u_1 u_2 (1 - \cos\chi) & u_2 \sin\chi + u_1 u_3 (1 - \cos\chi) \\ u_3 \sin\chi + u_1 u_2 (1 - \cos\chi)) & 1 + (u_2^2 - 1)(1 - \cos\chi) & -u_1 \sin\chi + u_2 u_3 (1 - \cos\chi) \\ -u_2 \sin\chi + u_1 u_3 (1 - \cos\chi) & u_1 \sin\chi + u_2 u_3 (1 - \cos\chi) & 1 + (u_1^2 - 1)(1 - \cos\chi) \end{pmatrix} \tag{5.54}$$

5.6 Symmetry averaging in practice

In most crystals, there is both crystallographic and non-crystallographic symmetry. In order to describe the action of crystallographic and non-crystallographic symmetry, the

terms "monomer" and "oligomer" are used here as follows. In a crystal, a set of *oligomers* make up the unit cell. The oligomers occupy compact domains which are related to each other by crystallographic symmetry. Each oligomer is made up of *monomers*, where each monomer is related to the other monomers in the same oligomer by NCS. Both monomers and oligomers are compact structures. The NCS projection operation is then to apply the NCS averaging to each of the oligomers. This can be done efficiently as follows

1. Apply the NCS averaging over the pixels in one of the oligomers.

2. Use the crystallographic symmetry to set the values for the pixels in the other oligomers.

3. Use the original values for pixels not in any of oligomers.

The tryptophanase protein from *proteus vulgaris*, entry 1ax4 in the PDB [55] is used as an example. It has order 4 crystallographic $P2_12_12_1$ symmetry and 4-fold NCS, and is shown in Fig. 5.5(a), where the four oligomers are shown in different colours (red, blue, yellow and green). The NCS consists of 4 mutually perpendicular 2-fold rotation axes as shown in Fig. 5.5. Symmetry averaging for this molecule proceeds as follows.

Step 1: Apply the NCS averaging over the pixels in one of the oligomers

Any oligomer can be used, and in this case the NCS averaging is applied over the red oligomer. Fig. 5.5(b) shows the red oligomer and Fig. 5.5(c) shows Fig. 5.5(b) with the NCS axes shifted to the centre of the unit cell for a clearer depiction. The centre of the NCS for the red oligomer is at $\mathbf{d} = (0.5, 59.1, 60.3)$Å. The three 2-fold rotation NCS axes labeled by the three colours black, magenta and cyan and are shown as the three lines. The orientation angles in degrees for the three NCS axes are

$$\text{axis } a: \quad \text{magenta} : (\phi_a, \psi_a, \chi_a) = (-89.45090, 103.76575, 180) \tag{5.55}$$

$$\text{axis } b: \quad \text{black} : (\phi_b, \psi_b, \chi_b) = (-11.91404, 46.58171, 180) \tag{5.56}$$

$$\text{axis } c: \quad \text{cyan} : (\phi_c, \psi_c, \chi_c) = (14.0137, -46.4560, 180). \tag{5.57}$$

Although there are three axes, only two of them are needed to apply the symmetry since the rotation about the third axis is equivalent to the rotation about the first two axes in succession. Using Eq. (5.54) and axes a and b, all four combinations of

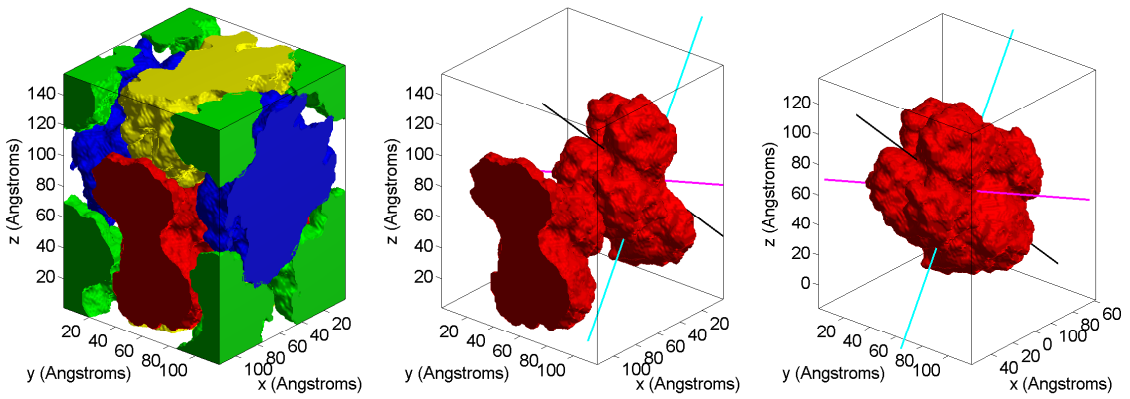


Figure 5.5 The tryptophanase protein from *proteus vulgaris* in the $P2_12_12_1$ unit cell. (a) Full protein in the unit cell, with each oligomer in a different colour, (b) the red oligomer and the NCS rotation axes, (c) the red oligomer with axes shifted to the centre of the unit cell.

application of the two 2-fold axis rotations gives the 4 NCS rotation matrices of

$$
\begin{aligned}
R_1^{NCS} &= \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \end{pmatrix} \\
R_2^{NCS} &= \begin{pmatrix} -0.9998 & -0.0181 & -0.0044 \\ -0.0181 & 0.8866 & 0.4622 \\ -0.0044 & 0.4622 & -0.8868 \end{pmatrix} \\
R_3^{NCS} &= \begin{pmatrix} 0.0102 & -0.2131 & 0.9770 \\ -0.2131 & -0.9550 & -0.2061 \\ 0.9770 & -0.2061 & -0.0552 \end{pmatrix} \\
R_4^{NCS} &= \begin{pmatrix} -0.0107 & 0.2313 & -0.9728 \\ 0.2624 & -0.9381 & -0.2259 \\ 0.9649 & -0.2577 & -0.0507 \end{pmatrix}.
\end{aligned} \tag{5.58}
$$

The NCS averaged value at each sample position **t** in the oligomer Q_{red} is given by

$$
\mathbf{x}'_{red}(\mathbf{t}) = \frac{1}{4}\sum_{k=1}^{4} x[(C_k(\mathbf{ts}))/\mathbf{s}] \quad \text{for } \mathbf{t} \in Q_{red} \tag{5.59}
$$

$$
= \frac{1}{4}\sum_{k=1}^{4} x[(R_k^{NCS}(\mathbf{ts}-\mathbf{d})+\mathbf{d})/\mathbf{s}] \quad \text{for } \mathbf{t} \in Q_{red}, \tag{5.60}
$$

where **s** is the sampling distance between pixels in Å, and the rotation matrices R_k^{NCS} are given in Eq. (5.58). The arguments of $x[\mathbf{t}]$ above are not sample points, so interpolation must be used.

Step 2: Use the crystallographic symmetry to set the values for the pixels in the other oligomers

Instead of using Step 1 to calculate the values of the samples in the other oligomers, it is more efficient to apply the crystallographic symmetry since this does not involve interpolation.

The operations to map the red oligomer to the other oligomers using the $P2_12_12_1$ crystallographic symmetry are

$$\begin{aligned}
\mathbf{x}'_{yellow}(t_1,t_2,t_3) &= \mathbf{x}'_{red}(-t_1+0.5L_1,-t_2,t_3+0.5L_3) \qquad \text{for } \mathbf{t} \in Q_{yellow} \\
\mathbf{x}'_{green}(t_1,t_2,t_3) &= \mathbf{x}'_{red}(-t_1,t_2+0.5L_2,-t_3+0.5L_3) \qquad \text{for } \mathbf{t} \in Q_{green} \\
\mathbf{x}'_{blue}(t_1,t_2,t_3) &= \mathbf{x}'_{red}(t_1+0.5L_1,-t_2+0.5L_2,-t_3) \qquad \text{for } \mathbf{t} \in Q_{blue}
\end{aligned} \tag{5.61}$$

where $\mathbf{L} = (L_1, L_2, L_3)$ is the size in pixels of the unit cell. The masks are given by

$$\begin{aligned}
Q_{yellow} &= \{\mathbf{t} : (-t_1+0.5L_1,-t_2,t_3+0.5L_3) \in Q_{red}\} \\
Q_{green} &= \{\mathbf{t} : (-t_1,t_2+0.5L_2,-t_3+0.5L_3) \in Q_{red}\} \\
Q_{blue} &= \{\mathbf{t} : (t_1+0.5L_1,-t_2+0.5L_2,-t_3) \in Q_{red}\}
\end{aligned} \tag{5.62}$$

The masks must be chosen such that they do not overlap. This is explained in further detail in Sec. 5.7.8.

Step 3: Use the original values for pixels not in any of oligomers

The combined NCS averaged image $\mathbf{x}'$ is then

$$\mathbf{x}'(\mathbf{t}) = \begin{cases}
\mathbf{x}'_{red}(\mathbf{t}) & \text{if } \mathbf{t} \in Q_{red} \\
\mathbf{x}'_{yellow}(\mathbf{t}) & \text{if } \mathbf{t} \in Q_{yellow} \\
\mathbf{x}'_{green}(\mathbf{t}) & \text{if } \mathbf{t} \in Q_{green} \\
\mathbf{x}'_{blue}(\mathbf{t}) & \text{if } \mathbf{t} \in Q_{blue} \\
\mathbf{x}(\mathbf{t}) & \text{if } \mathbf{t} \notin Q_{red} \cup Q_{yellow} \cup Q_{green} \cup Q_{blue}
\end{cases} \tag{5.63}$$

In practice only two masks are needed, the mask Q_{red} and the mask $Q_{red} \cup Q_{yellow} \cup Q_{green} \cup Q_{blue}$. Setting $\mathbf{x}'_{red}(\mathbf{t}) = 0 \ \forall \ \mathbf{t} \notin Q_{red}$, the usual crystallographic symmetry averaging is applied and the result multiplied by $q = 4$ to get the pixels values in $Q_{red} \cup Q_{yellow} \cup Q_{green} \cup Q_{blue}$.

5.7 Finding the NCS parameters

In order to apply an NCS projection, it is necessary to know the position and orientation of the rotation axes and the region Q over which the NCS symmetry applies. The orientation of the NCS axes can be found reasonably straightforwardly from the self-rotation of the Patterson function [2] but determining the position of the axes and the region of NCS symmetry is more difficult. In general, some additional information is needed to determine the position and shape of the NCS region. The exception is in crystallography of spherical viruses, where the spherical nature of the virus and the high degree of symmetry allow the NCS parameters of virus crystal to be easily found without any additional information.

In most crystals, the NCS applies to all the protein regions with the exception of the interfaces between oligomers, which must exist to give the oligomer structural integrity. The support (molecular envelopes) can therefore be used as the NCS mask Q. A conservative approach would be to use a smaller region Q such that the NCS symmetry must apply within this region, but in practice this approach would lead to regions Q that are quite small, significantly reducing the power of the NCS constraint.

Various standard and possible techniques for determining the NCS parameters are described below. Generally a combination of these techniques must be used to find the NCS parameters.

5.7.1 Rotation function

The rotation function is a tried and tested method for finding the orientation of the NCS rotation axes [10]. Consider a crystal $\mathbf{x}$ with crystallographic symmetry C^{CS} and non-crystallographic symmetry C^{NCS}. If a rotation R is applied to $\mathbf{x}$, its Patterson function is also rotated in an identical manner. Note that the Patterson function can be calculated directly from the diffraction data. The region of the Patterson function near the origin is denoted U and consists primarily of intra-oligomer vectors, and is a superposition of the intra-molecular vectors of the oligomers.

The rotation function Y is defined as the integral of the overlap function of the Patterson function and its rotated version

$$Y(\phi,\psi,\chi) = \int_U P(\mathbf{t})P(R(\phi,\psi,\chi)\mathbf{t})d\mathbf{t}, \tag{5.64}$$

The rotation function gives a large value at values of (ϕ,ψ,χ) which correspond to a rotation which maps one oligomer to an NCS-rotated version of itself. Therefore, a search of the Patterson function gives the orientations of the NCS axes. Some complications result with interpolation when calculating the rotated Patterson func-

tion, but the above shows the essence of the technique.

If **x** has a crystallographic symmetry and there is an NCS rotation axis oriented at (ϕ_1, ψ_1, χ_1) for one of oligomers, it is possible to directly calculate the NCS rotation axes (ϕ_2, ψ_2, χ_2) for the other oligomers using the crystallographic symmetry. In this way, the crystallographic symmetry can be used to average the rotation function. This method is called the *locked rotation function* and increases the signal to noise ratio of the rotation function.

Alternatively, all the Patterson functions which are related to each other by the crystallographic symmetry can be overlapped. The Rotation function is then given by

$$Y(\phi, \psi, \chi) = \int_U P(\mathbf{t})P(R_0^{CS}\mathbf{t})P(C_1^{CS}R\mathbf{t})...P(R_{q^{CS}}^{CS}R\mathbf{t})d\mathbf{t} \tag{5.65}$$

where R_k^{CS} are the rotation matrices for the q^{CS}-fold crystallographic symmetry.

5.7.2 Translation function

The translation function compares inter-oligomer Patterson peaks to find the distances between oligomers. The distance can then be used to find the position of the oligomers, or translation of the NCS axes, in the crystal. The usual translation function [56] assumes knowledge of an approximate structure which is then used to calculate a Patterson function. The calculated Patterson function is then compared with the observed Patterson function to find the distance between oligomers, which gives the relative positions of the NCS axes.

In an analogous way to the rotation function, it is possible to compare inter-oligomer Patterson peaks corresponding to the distance between pairs of oligomers which have the same relative orientation to find their positions in the unit cell. Let there be three oligomers O_1, O_2, and O_3, with NCS axis centres at $\mathbf{d}_1$,$\mathbf{d}_2$ and $\mathbf{d}_3$.

Each point t_1 in O_1 can be mapped to its identical point t_2 in O_2 with

$$t_2 = R(t_1 - \mathbf{d}_1) + \mathbf{d}_2 \tag{5.66}$$

and each point t_2 in O_2 can be mapped to its identical point t_3 in O_3 with

$$t_3 = R(t_2 - \mathbf{d}_2) + \mathbf{d}_3. \tag{5.67}$$

where R is the rotation matrix for both operations. Then the Patterson function at $\mathbf{d}_{12} = \mathbf{d}_1 - \mathbf{d}_2$ consists of inter-oligomer vectors between O_1 and O_2 and will be similar to a rotated (by R) version of the Patterson function at $\mathbf{d}_{23} = \mathbf{d}_2 - \mathbf{d}_3$. The

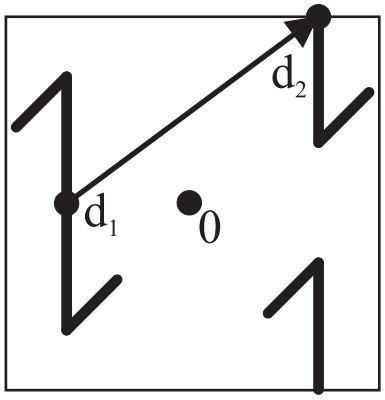


Figure 5.6 Parallel crystallographic and non-crystallographic axes.

translation function is then calculated as

$$T(\hat{\mathbf{d}}_{12}, \hat{\mathbf{d}}_{23}) = \int_U P(\mathbf{t} - \hat{\mathbf{d}}_{12}) P(R(\mathbf{t} - \hat{\mathbf{d}}_{23})) d\mathbf{t}. \tag{5.68}$$

The translation function is large when $\hat{\mathbf{d}}_{12} = \mathbf{d}_{12}$ and $\hat{\mathbf{d}}_{23} = \mathbf{d}_{23}$ which in principle allows d_{12} and d_{23} to be determined. The major problem with this method is that as there are many stray Patterson peaks and the signal to noise ratio is poor. Furthermore, the area of integration U is difficult to define. In practice, the translation function is not very effective and finding the position of the NCS axes is difficult.

5.7.3 Parallel NCS and CS axes

If a NCS rotation is similar to a crystallographic symmetry rotation, i.e. their axes are parallel or close to parallel, then two oligomers are oriented very similarly, so there is a large peak in the Patterson function corresponding to the distance between the two oligomers. This can then be used to find the position of the NCS axes.

A 2D example of parallel crystallographic and non-crystallographic axes is shown in Fig. 5.6. The crystallographic symmetry consists of a 180° rotation about the origin followed by a shift of half the unit cell height. The NCS for each of the two oligomers consists of a 180 degree rotation around points $\mathbf{d}_1$ and $\mathbf{d}_2$ as shown in the figure. The Patterson function has a large peak at the position $\mathbf{d}_1 - \mathbf{d}_2$ which is where the two oligomers overlap, giving an estimate of $\mathbf{d}_1 - \mathbf{d}_2$.

5.7.4 Parameterizing the envelope

At low resolution a protein can be approximated by a constant electron density, so by parameterizing the shape of the oligomer envelope with a small number of parameters, the Fourier magnitudes of the parameterized envelope in the unit cell can be calculated and compared with the diffraction data, allowing a rough estimate of the envelope to be found by finding the parameters that give the best match. If the position of the oligomers in the unit cell is not known, the position can also be included as another three parameters in the search, which greatly increases the computational requirements, but nevertheless should work in principle. This method is commonly used in virus crystallography since the virus molecule is approximately a spherical shell and can therefore be easily parameterized by an inner and outer radii, and the position of the virus NCS axes is generally known.

Success has also been found in approximating the protein with a number of large spheres and applying direct methods techniques [57].

5.7.5 Known oligomer support

If the support (shape and orientation) of the oligomer is known, for example by electron microscopy, a stearic search can be used to find the position **d** of the oligomer in the unit cell. The idea is that possible positions of the molecule are those that do not result in overlap with the crystallographic symmetry generated copies. For each position of the mask in the asymmetric unit, all the crystallographic symmetry operations as shown in Eq. (5.62) are applied to generate all masks in the unit cell. The total amount of overlap between the masks, i.e. $|Q_{red} \cap Q_{yellow} \ldots \cap Q_{blue}|$ is then determined and the most likely position for the oligomer is that with the minimum overlap.

If, for example, there is 4-fold NCS consisting of three orthogonal 2-fold rotation axes as in the example above, then there are a total of 6 possible orientations of the oligomer support which satisfy the NCS, and all 6 orientations need to be tested using a stearic search as described above. An example of this approach is described in Sec. 6.5.1.1. Note that if the support of the oligomer is known, the position and orientations of the NCS axes can be inferred from the symmetries of the support.

5.7.6 Finding the NCS axes position from an electron density estimate

If the oligomer support or electron density is known, then the origin of the NCS axes can be taken as the centre of mass of the oligomer

$$\bar{t} = \sum_{\mathbf{t}} |\mathbf{t}| x[\mathbf{t}], \tag{5.69}$$

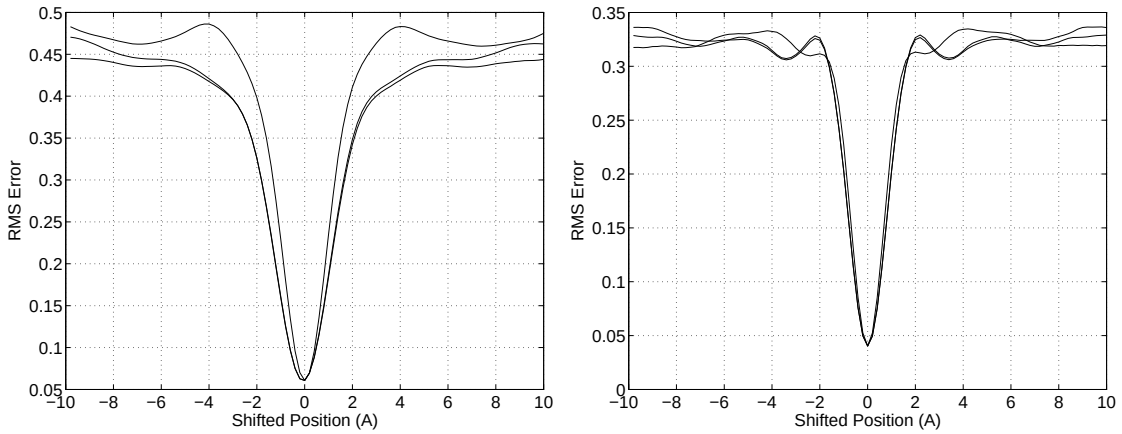
where the summation is over all samples of the image. If the estimate is noisy, this method may not work well.

An alternative method is to search possible positions of the NCS axes, apply the NCS symmetry averaging operation, and calculate the difference (error) between the estimated electron density before and after NCS averaging. The best NCS position is that with the smallest error. Using the protein from Sec. 5.6, the difference as a function of position of the NCS origin along the x-axis is shown in Fig. 5.7. The error is well behaved (monotonic) to around 3Å from the true position, so an iterative gradient search can be used to locate the best origin. At each iteration, the error with the current NCS origin is calculated, as well as the three errors with the NCS axes shifted one step length in each of the 3 unit cell axes directions, with the initial step length set to 1 grid spacing. If the error decreases along an axes, then the next iteration moves in that direction, otherwise, it moves in the opposite direction along the same axes. The step length is then decreased by 0.7 and the procedure repeated until there is no change. This method was found to converge rapidly and to improve the NCS origin.

Since the estimates of the electron density are from the IPA which enforces NCS symmetry, the estimates themselves are heavily biased towards the current NCS position, especially if the usual estimate $P_I T_F \mathbf{x}$ is used. One way to decorrelate the estimate from the NCS position is to use the relaxed projection from the iterate $T_F \mathbf{x}$ as the estimate for the NCS axes position refinement. Note that some biasing of the new NCS position towards the current estimate is good as it prevents the new NCS axes position from moving too far from the current NCS position.

5.7.7 Known molecular support or a low resolution electron density

Using, for example, a solvent contrast method as described in Chapter 4, it may be possible to find the envelope or low resolution electron density in the unit cell. The position of the NCS axes can usually be found from this envelope. The support must then be partitioned into supports for each of the oligomers.



(a) (b)

Figure 5.7 Error for shifted NCS axes position using (a) the true electron density, and (b) the reconstructed electron density.

For each grid point **t** in the asymmetric unit, the position of the q_{NCS} NCS-related points are computed assuming that **t** is in each of the q_{CS} oligomers, resulting in q_{CS} sets of q_{NCS} points. The value of the low resolution electron density at these points can then be found by interpolation. The set of q_{NCS} values for the correct oligomer should be nearly identical.

A method to find the oligomer masks is then to compute the standard deviation for each of the q_{CS} sets of q_{NCS} NCS related points, giving the values $\sigma_1, ..., \sigma_{q_{CS}}$ for each pixel **t** in the image. Then a "probability" metric p for the pixel to be in the first oligomer is given by

$$p_1[\mathbf{t}] = \sigma_1 / \left(\sum_{j=1}^{q_{CS}} \sigma_j \right). \tag{5.70}$$

Since each oligomer is compact, the $p_1[\mathbf{t}]$-image is low-pass filtered, and a threshold is used to select the pixels belonging to the first oligomer. Crystallographic symmetry is used to generate the supports of the other oligomers, and overlaps can be resolved by removing them from one or both oligomer masks.

This method can be used as part of the iterations to refine the NCS mask. The relaxed projection onto the Fourier magnitude projection should be used to minimize the correlation between the current and estimated oligomer masks.

5.7.8 Determining the NCS support if the NCS axes are known

In the case of a spherical virus, the position and orientation of the NCS axes is generally known. It is then possible to estimate an NCS mask as follows. First, each pixel in the unit cell is assigned to its nearest NCS axes centre. Let all the

pixels assigned to one of the oligomers be denoted Q_1', with the NCS operations related to that oligomer denoted C_1^{NCS}. Note that Q_1' is actually a full asymmetric unit of the unit cell. Then the NCS mask estimate Q_1 can be found by selecting the pixels for which all the NCS related points are in Q_1', i.e.

$$Q_1 = \{\mathbf{t} : \ C_1^{NCS}\mathbf{t} \in Q_1', \ \forall \text{ elements of } C_1^{NCS}\}. \tag{5.71}$$

This method works well if there is minimal interlocking between oligomers, which is generally the case for spherical viruses. If the oligomers are interlocked, then the method of the previous section could be used, but a low resolution electron density must first be found.

5.8 Conclusions

Symmetry properties of images provide a constraint that is useful in image reconstruction. Symmetry can be divided into global or local, or crystallographic and non-crystallographic, symmetry. Non-crystallographic symmetry provides additional information in the case of reconstruction from Fourier magnitudes. For a continuous image, the symmetry projection is equivalent to symmetry averaging, but the projection requires the use of interpolation for sampled images. Sinc interpolation provides perfect interpolation for bandlimited images, but the computational requirements are too high for use in IPAs. Linear interpolation can be used in a projection in order to keep the computational load manageable, but causes the image to be smoothed. The definition of a symmetry constraint set for a sampled image depends on the method of interpolation that is used. The definition is straightforward for sinc interpolation of bandlimited images. For linear interpolation the symmetry constraint set is defined here in terms of the maximum change in the image on the second application of the symmetry projection.

Various methods to find the parameters of the symmetry are described. The orientation of the NCS axes can be found from the Patterson function, but finding the position of the axes remains a problem. Most methods for finding the translational position of the axes require the collection of some additional information or a similar crystal.

Chapter 6

Phasing Crystal Diffraction Data using Symmetry

6.1 Introduction

Despite recent advances, determination of the structures of large and complex macromolecular structures from crystal x-ray diffraction data can sometimes be problematic as a result of the experimental difficulties of obtaining sufficiently accurate initial phase information. Therefore, despite the power of modern methods for macromolecular crystallography, *ab initio* phasing techniques that do not require any experimental phase estimates would offer considerable advantages in some cases. Outside the regimes where direct methods or molecular replacement methods are effective, *ab initio* phasing will require the use of ancillary structural information to compensate for the lack of phase information. In the absence of the structure of a homologous molecule, the main structural constraint available is that of symmetry.

Non-crystallographic symmetry as described in the previous chapter has long been recognized as providing additional phasing information in macromolecular crystallography. It was first developed in Fourier space by Rossmann and Blow (1963), Main and Rossmann (1967), and Crowther (1969) [54, 58, 59], but proved to be of little utility until the development by Brigcogne (1974) [60] of image domain NCS averaging. The incorporation of NCS is now a routine component of electron density modification algorithms for phase refinement and extension [53]. In these algorithms, a non-crystallographic symmetry constraint is incorporated by symmetry averaging, i.e. for each grid point, locating the other symmetry-related points, determining the electron density at these points by interpolation, and then replacing the electron density of the original grid point by the average value over the symmetry related points. This is done at each cycle of electron density modification,

the effect being to enforce the non-crystallographic symmetry on the electron density at each cycle. This approach has been highly successful in refining phases and extending them to higher resolution; however, its success does depend on the initial phases being reasonably accurate, i.e. the electron density prior to averaging being reasonably accurate. If this is not the case then the electron density does not improve after each averaging cycle and does not converge to the correct solution.

These electron density modification techniques correspond to the simplest kind of projection algorithm that have poor global convergence properties must therefore be started close to the solution. As a result, they have been used almost exclusively for phase refinement and extension from an initial set of experimental phases. It is possible, therefore, that the general failure of electron density modification algorithms for *ab initio* phasing is due to the simplicity of the algorithms used.

There have been attempts, however, with some limited success, to apply these algorithms to *ab initio* phasing, i.e. starting with no initial experimental phase estimates. The most successful of these has been with icosahedral viruses due to the high degree of structural redundancy resulting from a high degree of NCS [61], and the ease of estimating the molecular envelope and the location of the NCS axes. One of the first applications of this approach was to canine parvovirus (CIV) [62]. An initial phase set to 20Å resolution was constructed based on a spherical shell corresponding to the viral dimensions and these were refined using the measured diffraction magnitudes while applying the 60-fold NCS using classical density modification, followed by extension to 9Å resolution. However, extension to higher resolution was not possible and it was necessary to use isomorphous replacement phases to solve the structure. Similarly, attempts to solve the structure of *nudaurelia capensis* ω virus, which also has 60-fold non-crystallographic symmetry, starting from phases based on a spherical shell model could not be extended to sufficient resolution to obtain an interpretable electron density [63], and heavy atom data and partial atomic model building were required to solve the structure. For lower degrees of NCS, such as 5-fold NCS, some progress has been made with *ab initio* phasing with Naitow et al. (1999) and Taka et al. (2005) [64, 65] able to phase icosahedral viruses with 5-fold NCS *ab initio* using experimental data by starting from a few different initial positions, *ab initio* phasing at low resolution, selecting the best model using a combination of real-space metrics and manual analysis, and then pixel-by-pixel phase extension using Rayment weighting and masking techniques at each iteration to achieve a resolution between $4.3-10$Å. The algorithm used was equivalent to the error reduction algorithm described in Sec. 1.3.3.1. Use of more sophisticated projection algorithms is examined here.

In this chapter the application of iterative projection algorithms that incorporate a non-crystallographic symmetry constraint to macromolecular crystallography is investigated. The algorithm synthesizes the NCS projection operators described in

Ch. 5 with the Fourier magnitude projection to reconstruct macromolecular electron densities from (undersampled) crystal diffraction amplitudes.

The question of uniqueness of the solution with symmetry, envelope and Fourier magnitude constraints is addressed in the next section. The IPA used is briefly reviewed in the following section. The methods are applied to a symmetric icosahedral virus and a symmetric protein in the next two sections. Concluding remarks are made in Sec. 6.6.

6.2 Uniqueness

It is important when contemplating *ab initio* structure determination to first consider whether the given data and constraints are sufficient to provide a unique solution. If this is not the case then there is little point in pursuing algorithms for structure determination since any effective algorithm may find one of a multitude of incorrect solutions. Uniqueness for the case of measured crystal diffraction magnitudes and known molecular envelope and NCS constraints is considered here.

Arnold and Rossmann (1986) [66] considered the effect of molecular envelope volume and NCS order and defined a quantity they called "phasing power" that allows one to compare the signal-to-noise improvement provided by different sets of constraints. However, the phasing power does not provide any information on the expected uniqueness of the solution. Millane (1993) [67] studied uniqueness properties for macromolecular crystallography that considered the shape of the support region (molecular envelope) and the order of the NCS, and gave a parameter that could be related to uniqueness of the solution. Recently, Elser and Millane (2008) [68] considered uniqueness of the phase problem for continuous diffraction data from isolated molecules. They defined the "constraint ratio", denoted Ω, as the ratio of the number of independent data (diffraction magnitudes) divided by the number of independent object (electron density) parameters. For $\Omega > 1$ a unique solution is to be expected. In practice, an additional margin is required to account for noise in the data. The constraint ratio depends only on the shape and dimensionality of the support region. They showed that, in three dimensions, for support regions that are convex and centrosymmetric (expected to be approximately the case for many molecular envelopes), $\Omega = 4$. The problem is therefore highly overconstrained for continuous (non-crystalline) diffraction data.

These results can be extended to the case of crystal diffraction data. For crystal diffraction data the autocorrelation of the molecular support is replicated with the translational symmetry of the crystal and aliasing occurs. Since the molecular envelope generally extends to the edges of the unit cell for crystal integrity, the autocorrelation almost always completely fills the unit cell. The volume of the auto-

correlation containing independent coefficients (data) is therefore half the volume of the unit cell (as a result of Hermitian symmetry), so that $\Omega = 0.5$, i.e. the usual crystal diffraction case is highly non-unique. If the known support of the molecular envelope occupies a fraction f of the unit cell, the number of object parameters is reduced by a factor of f. If the R-fold symmetry applies over the entire support then the number of object parameters is reduced by an additional factor of R. Putting all this together gives

$$\Omega = \frac{R}{2f}, \tag{6.1}$$

with Ω required to be > 1 for uniqueness. If the worst case of f being fairly close to unity is assumed, then generally $R > 2$ is required, i.e. more than 2-fold NCS. Therefore, considering the effects of noise in the data, missing data, and other uncertainties, it seems reasonable that at least 3-fold NCS, but maybe not much more, should be sufficient to expect a unique solution for *ab initio* phasing in macromolecular crystallography.

6.3 Algorithm

The Fourier magnitude, support, and positivity projections described in Sec. 1.3.4 were used. The non-crystallographic symmetry projection was applied using linear interpolation as described in Sec. 5.3.5. The DM, ER and GHIO algorithms as described in Sec. 1.3.3 were used. For the HIO algorithm, 50 cycles of GHIO were also alternated with 5 cycles of ER, which is referred to here as the GHIOER algorithm. Since all of the projections maintain crystallographic symmetry, the crystallographic symmetry constraint can be enforced directly upon the iterate at the end of each iteration as described in Sec. 2.6.2.

Reconstructions were first performed at low resolution and the resolution then extended in steps as described in Sec. 2.7. At each resolution extension, the estimate with the lowest Fourier error metric was used as the initial iterate for the next resolution extension. The full grid was used at all resolutions, and the diffraction data were windowed with a Gaussian with a half height at the stated resolution by multiplying each Fourier magnitude by a Gaussian coefficient i.e.

$$M'[\mathbf{h}] = M[\mathbf{h}]\exp\left(\frac{(h_1/L_1)^2 + (h_2/L_2)^2 + (h_3/L_3)^2}{-E^2 ln(2)}\right). \tag{6.2}$$

where $M'[\mathbf{h}]$ denotes the windowed Fourier magnitudes, L_j denote the unit cell dimensions, and E is the half width of the Gaussian at half height, which is used here as the definition of resolution in Angstroms. A comparison between the Gaussian weighting method and the usual crystallographic bootstrapping method of zeroing out all frequencies above the stated resolution (E) is shown in Fig. 6.1. On the

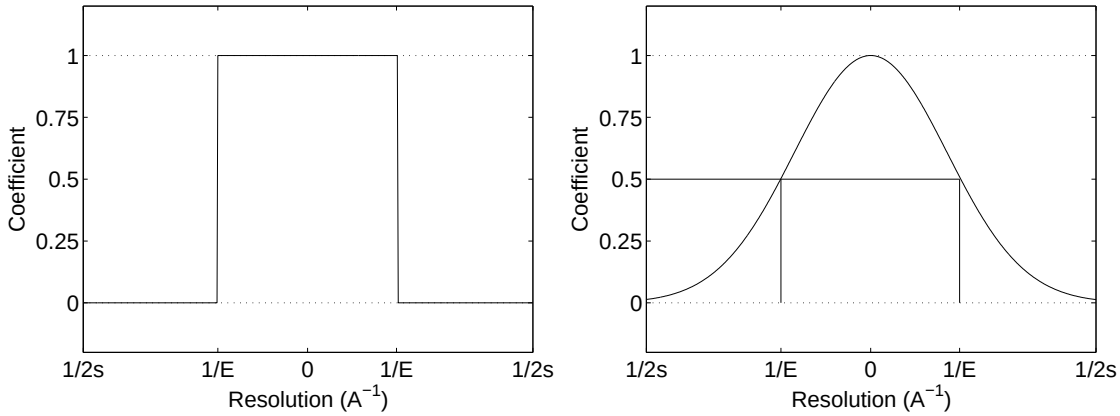


Figure 6.1 Bootstrapping coefficients for (a) the usual crystallographic method of zeroing out all Fourier magnitudes with resolution greater than E, (b) the Gaussian weighting method.

final extension, no windowing was used.

Application of the Gaussian weighting is a form of low-pass filtering which ensures that adjacent pixels in the electron density map are correlated. This improves the interpolation, but may result in the image no longer satisfying the symmetry as described in Sec. 2.7. After windowing, the new NCS mask Q' is given by

$$Q = \{\mathbf{t} : ((1 - Q) \odot S_g)[\mathbf{t}] = 0\} \tag{6.3}$$

where Q is the mask which is 1 where the symmetry applies and zero outside, S_g is the approximate support of the Gaussian, e.g. two standard deviations, and $((1 - Q) \odot S_g)[\mathbf{t}]$ denotes the value of $((1 - Q) \odot S_g)$ at $\mathbf{t}$. So if the bootstrapping resolution E is small so that the Gaussian impulse response is sufficiently narrow in the image domain, the symmetry will still apply over much of the image.

The advantage of bootstrapping is that the linear interpolation is more accurate at low resolution, so that the starting iterates at higher resolution are more accurate. The estimates will then still be useful at high resolution even if the bowl of attraction is small due to the approximate projections and the algorithm fails to make any more real progress towards the solution.

6.4 Reconstruction of an Icosahedral virus

A virus is an infection agent which uses the resources of its host cell to replicate. First discovered by Beijerinck in 1898 [69], it was not until the late 1970's that determination of the structures of complete viruses was possible [70]. Since then, the structure of many more viruses have been determined, but determination of

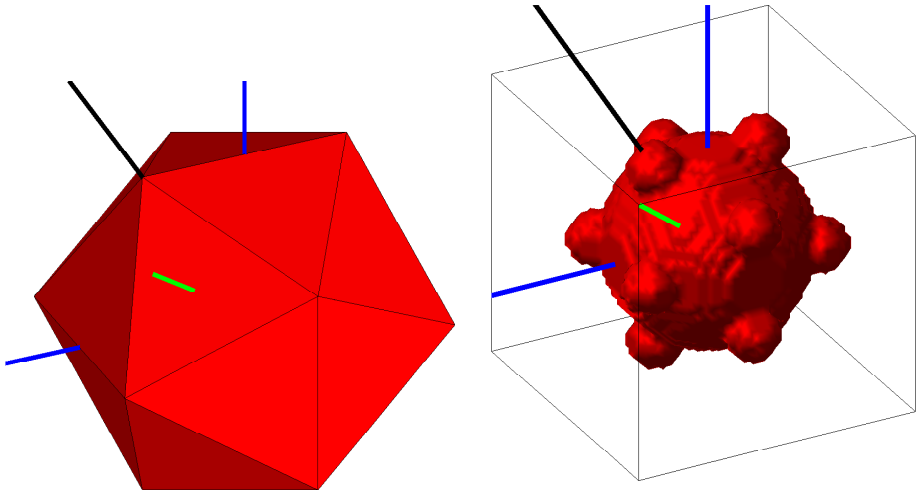
more complex viruses is still a challenge. Icosahedral viruses are made up of a spherical shell of proteins called the capsid, along with RNA or DNA and other disordered (i.e. different for each cell in the crystal) cell bodies in the interior [69]. The spherical capsid has a high degree of symmetry, most commonly icosahedral symmetry, which is a 60-fold point group symmetry. An icosahedron is a 20-sided platonic solid with triangular faces as shown in Fig. 6.2(a). It has 2-fold rotation axes through the centre of each edge from the centre of the icosahedron, 3-fold rotation axes through the centre of each face, and 5-fold rotation axes through each vertex.

When a virus with icosahedral symmetry is in a cubic unit cell then, depending on the setting of the virus, the resulting NCS can vary between 5-fold and 60-fold. A depiction of icosahedral symmetry with 5-fold NCS is shown in Fig. 6.2(b). The blue and green axes are 2-fold and 3-fold crystallographic rotation axes, and the black axes is a 5-fold NCS rotation axes. These 4 axes are the minimal set of axes required to generate the 60-fold icosahedral symmetry. In this case, the icosahedral 2-fold and 3-fold axes coincide with the crystallographic 2-fold and 3-fold axes so that only the 5-fold axes is non-crystallographic.

Due to the high degree of symmetry and the spherical nature of the virus molecules, the molecular envelope and position and orientation of the NCS axes are rather easily determined *a priori*, and as such, their structure determination has benefitted significantly from the use of NCS constraints. For cases with a high degree of NCS the degree of redundancy is high and phasing has been demonstrated starting with very little information. The problem is more difficult for lower degrees of NCS.

6.4.1 Methods

The algorithms described in Sec. 6.3 were tested using the experimental diffraction data from the melon necrotic spot virus [71], entry 2zah in the Protein Data Bank. The virus has icosahedral symmetry and packs in a cubic unit cell of dimensions $375 \times 375 \times 375$Å with $I23$ crystallographic symmetry, in which there is one virus molecule at the centre and one at the corner of the unit cell as shown in Fig. 6.3. This results in order 24 crystallographic symmetry and 5-fold non-crystallographic symmetry for a total of 120-fold point group symmetry from the two 60-fold symmetric icosahedral viruses. Experimental data between 267 and 2.8Å resolution are available, although only the data between 150Å and 2.8Å resolution were used. Using the atomic coordinates, standard crystallographic programs [72] were used to calculate a reference electron density map on a $268 \times 268 \times 268$ grid, corresponding to a grid spacing of 1.4Å and diffraction data with a resolution of 2.8Å. The simulations were carried out using MATLAB with some of the functions written in C. Simulation times were about 15 hours (including the calculation of many met-



(a) (b)

Figure 6.2 (a) An icosahedron (b) an object with icosahedral symmetry in a cubic unit cell resulting in 5-fold NCS.

rics) for approximately 500 iterations on a 2.5GHz machine with 3GB of memory.

6.4.1.1 Initial support choice and refinement

The molecular envelope (capsid) of an icosahedral virus is an approximately spherical shell. In a crystal, the region outside the virus is occupied by solvent, and the region inside the shell is occupied by solvent and possibly disordered RNA. The true average electron densities outside and inside the capsid are denoted ρ_{s1} and ρ_{s2}, respectively (in general, $\rho_{s1} < \rho_{s2}$). The support projection P_S is then given by

$$P_S x[\mathbf{t}] = \begin{cases} x[\mathbf{t}] & \text{if } \mathbf{t} \in V_S, \\ \rho_{s1} & \text{if } \mathbf{t} \in V_1, \\ \hat{\rho}_{s2} & \text{if } \mathbf{t} \in V_2 \end{cases} \tag{6.4}$$

where V_1 denotes the region outside the capsid, V_2 denotes the region inside the capsid, and V_S denotes the region of the capsid. A cross section of the capsid showing the three areas is shown in Fig. 6.4. Note that although ρ_{s1} is generally known quite accurately, ρ_{s2} is generally known only approximately. Therefore, an estimated value for ρ_{s2}, $\hat{\rho}_{s2}$, is used in the projection.

The initial support at the lowest resolution is a spherical shell with inner and outer radii of 210Å and 340Å, respectively. These approximate dimension can generally be determined using, for example, electron microscopy or solution scattering coupled with information provided by the molecular weight, space group and unit

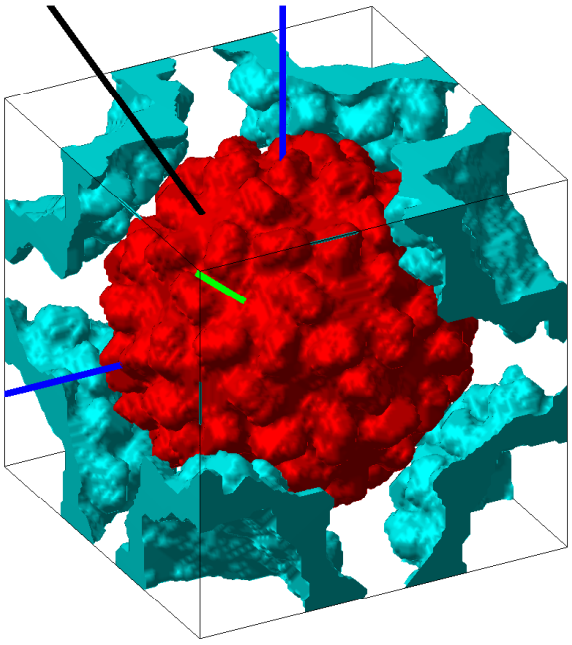


Figure 6.3 The melon necrotic spot virus at 30Å. The icosahedral virus sits in a unit cell with I23 symmetry. The blue and green axes are 2-fold and 3-fold crystallographic rotation axes, and the black axes is a 5-fold NCS rotation axes.

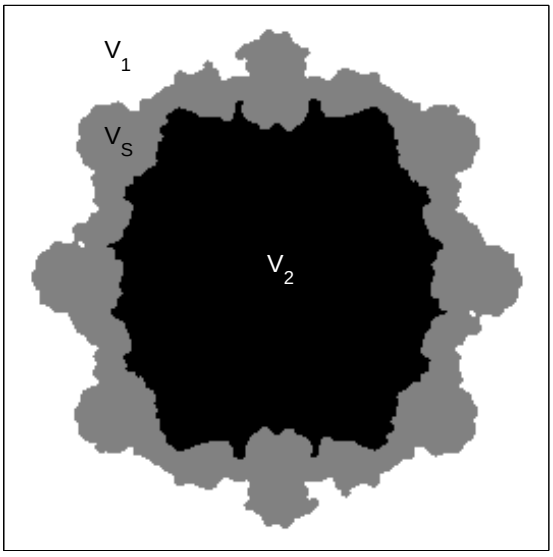


Figure 6.4 Partition of the virus into V_1, V_2 and V_S for the application of the symmetry constraint.

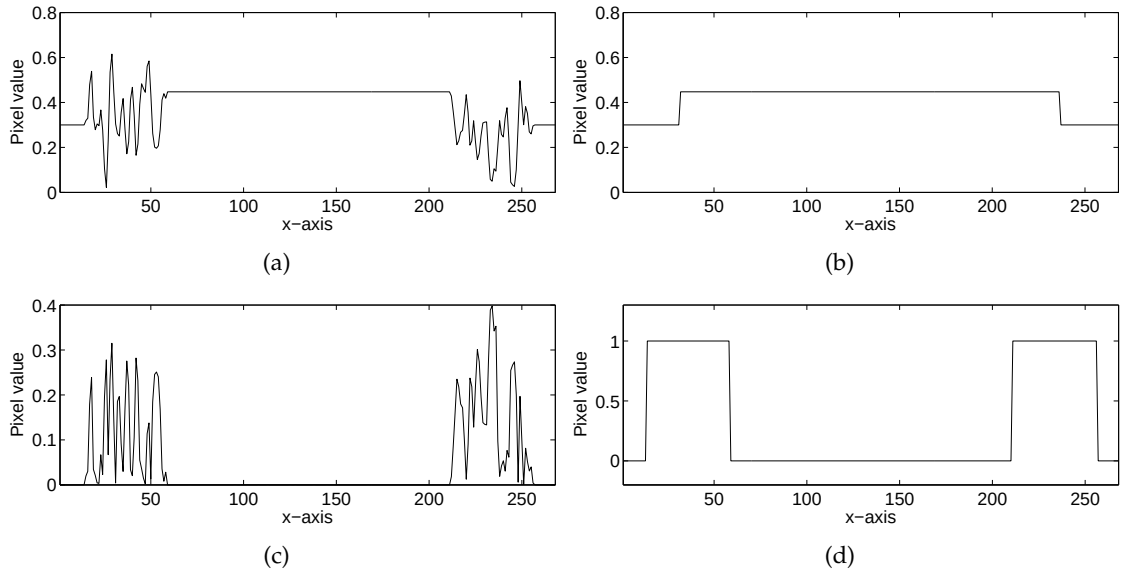


Figure 6.5 Cross section through the centre of a reconstructed virus showing the support partition technique. (a) Reconstructed electron density, (b) Spherical mask with values ρ_{s1} outside and $\hat{\rho_{s2}}$ inside, (c) the absolute difference between (a) and (b), (d) the result of smoothing and thresholding (c).

cell dimensions. The molecular envelope was refined as the resolution increases. At the beginning of each resolution step, the electron density map is partitioned into three regions, the capsid, the interior of the capsid, and the exterior as follows. First, a spherical mask is generated with a radius equal to the average of the inner and outer radii of the initial estimate (275Å) and the electron density set to ρ_{s1} outside the mask and $\hat{\rho_{s2}}$ inside. This mask is subtracted from the current electron density estimate, the absolute value taken, and the result is smoothed by multiplying the corresponding structure factors with a Gaussian window with a half height at 10Å resolution. The new map is then expected to be zero outside V_S and have large values inside V_S. The $1.1fN$ samples of the electron density with the largest values are then taken to define the support of the virus V_S, where f is the fraction of the unit cell filled by the support. The regions V_1 and V_2 are then easily determined. The value of $\hat{\rho_{s2}}$ is updated at the end of each resolution step by setting it to the average of the values of the electron density samples in the new region V_2. A diagram of the section through the middle of the virus as the above procedure is carried out is shown in Fig. 6.5.

6.4.1.2 NCS region

The position and orientation of the 5-fold NCS axis is fixed in this space group and passes through the body diagonals of the unit cell. The NCS averaging is applied using linear interpolation to keep the computational complexity low, and was effective because of the fine grid spacing used. The Gaussian window serves

another purpose here by keeping adjacent pixels correlated to maintain stability of the linear interpolation.

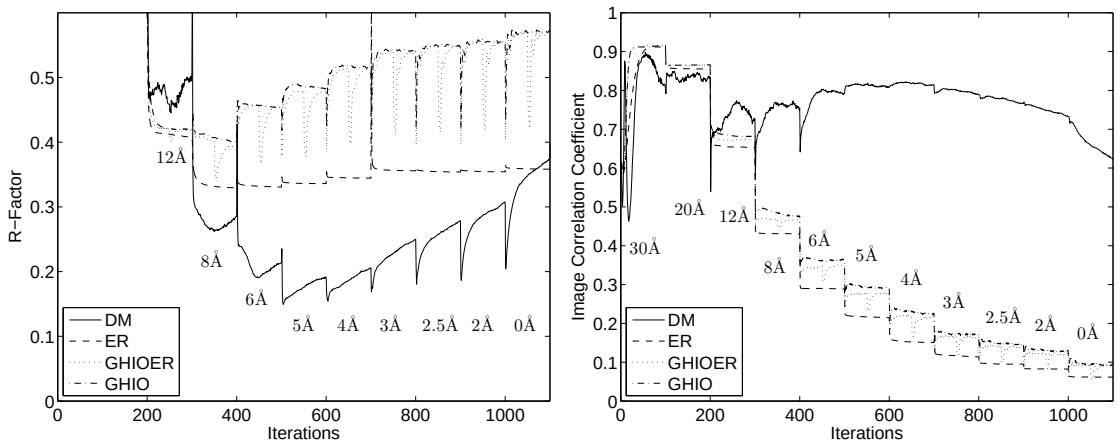
It can be inferred from the I23 symmetry that there are two virus particles in each unit cell, with one virus particle in the centre and one at the corner. Since the position of the NCS axes are known, the region over which the NCS symmetry applies can be determined as described in Sec. 5.7.8, where each grid point in the unit cell is assigned to one of the virus particles in the unit cell based on the distance to the centre of the virus particles. The NCS averaging is applied to all grid points assigned to one virus such that all grid points used in the interpolated averaging operation are assigned to the same virus.

6.4.2 Results

The ER, GHIO with $\beta = 0.7$ and DM with $\beta = 0.9$ algorithms were used. For the GHIO algorithm, 50 cycles of GHIO were also alternated with 5 cycles of ER, which is referred to here as the GHIOER algorithm. Three different starting electron densities were used for each simulation, but the algorithms gave very similar results regardless of initialization. The value of ρ_{s1} was set to 0.3 and the initial value of $\hat{\rho}_{s2}$ was set to 0.45. The inner and outer radii of the initial molecular envelope shell were set to 210Å and 340Å, respectively.

Progress of the reconstruction is presented using the R-factor and image correlation coefficient, although other metrics were monitored as well. The estimate at each iteration is given by $\hat{\mathbf{x}} = P_I T_F \mathbf{x}$ as described in Sec. 2.8.3. The four algorithms were first run with resolution extensions at 30, 20, 12, 8, 6, 5, 4, 3, 2.5, 2 Å and 200 iterations at each resolution to facilitate simple comparisons between the algorithms. On the final extension with a $268 \times 268 \times 268$ grid, no windowing was used.

The error metrics versus iteration are shown in Fig. 6.6 for the four algorithms. Inspection of the figure shows that while all the algorithms make some progress towards a solution at up to about 8 Å resolution, only the DM algorithm is able to find a good solution. The dip in the middle of the GHIOER algorithm is from the few iterations of ER. The ER algorithm stagnates in a local minima very quickly as can be seen by the horizontal lines in the error metrics. The GHIO algorithms are better able to explore the space, but are unable to find the correct solution. Even at low resolution the weights for the higher Fourier magnitudes are still significant, so that the high resolution phases are refined to some degree even at the low resolution stages. So when each resolution extension is carried out, convergence for the DM, GHIO, and GHIOER algorithms is rapid, after which the iterates begin to diverge from the solution due to noise and the error metrics rise. Note that the estimate for the next bootstrap is taken when the error is at the minimum from the previous bootstrap, so the divergence from the solution is immaterial. The relevant



(a) (b)

Figure 6.6 Error metrics vs Iteration (a) R-factor vs Iteration (b) Image Correlation Coefficient vs Iteration.

R-factors are therefore those at the dips in the curve.The ER algorithm is less efficient at converging after each bootstrap iteration, and this can be seen by the error metrics slowly falling and then stagnating after each resolution extension, since the ER algorithm does not search the space like the other three algorithms.

To facilitate comparison between algorithms in the error metrics versus iterations plots, a constant number of 100 iterations was run for each bootstrap in Fig. 6.6. In practice this is unnecessary, and the resolution extension can be carried out when the R-factor starts to increase. The reconstruction using the DM algorithm was repeated and the resolution extended when the mean R-factor of the last 10 iterations is greater than the mean R-factor of the preceding 10 iterations, i.e.

$$\text{extend resolution if } \sum_{k=n}^{n-9} R(k) < \sum_{k=n-10}^{n-19} R(k), \tag{6.5}$$

where $R(k)$ denotes the R-factor of the k^{th} iteration and n is the current iteration. The result is shown in Fig. 6.7 and the number of iterations required for the solution is reduced from about 1100 to 550 without any change in the results. When the R-factor is greater than 0.4, the algorithm is nowhere near convergence and so the resolution is not extended even if the R-factor increases, i.e. Eq. (6.5) is applied only if $R(k) < 0.4$. It can be seen in Fig. 6.7 that the algorithm takes longer to converge at the low resolution extensions. This is because the iterates are still far from the solution. After the algorithm is in the vicinity of the solution the small changes at each resolution extension means that convergence is rapid.

A common error plot in crystallography is to plot the error metrics versus resolu-

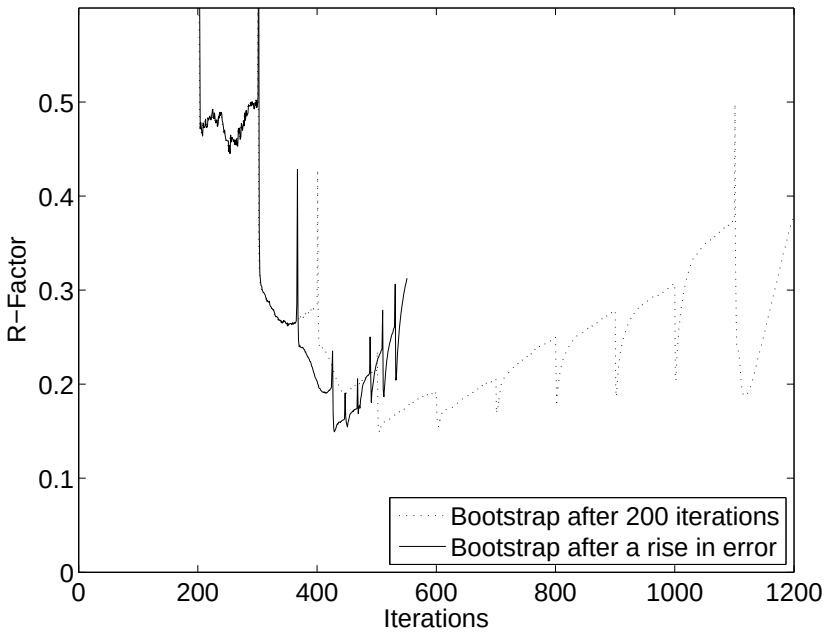
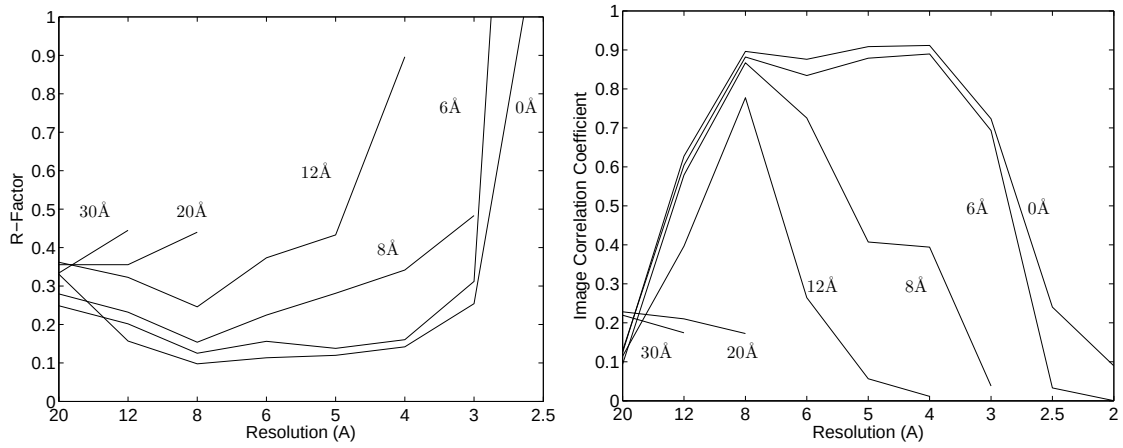


Figure 6.7 R-factor vs Resolution with bootstrapping when a rise in error is detected.

tion for each resolution extension. At each resolution extension, the estimate of the solution is found and filtered with a series of bandpass filters (resolution shells). Using bandpass filters results in what is known as a "shell-by-shell" plot, and an alternative method is to use a series of low pass filters instead. The error metric after each bandpass filter, i.e. at each resolution, is computed and plotted to create a line plot of the error metrics as a function of resolution for each estimate. Plotting a line for each resolution extension allows the improvement in resolution as the bootstrapping proceeds to be observed. Using the results from the DM algorithm, the error metrics versus resolution plots are shown in Fig. 6.8. As the bootstrapping proceeds, the improvement of the metrics at the higher resolutions can clearly be seen. The poor error metrics at low resolution are an artifact of the error metric calculation. The published structure used to calculate the "true" electron density map does not include the solvent, and so an estimated flat solvent level is used, which is not necessarily correct. In practice, this is not a problem since the zero frequency term is not measured, and so the electron densities will shift by a constant amount so to correspond to the given solvent density level.

Central slices of the reconstructed capsid electron density at 5Å resolution are shown for each algorithm in Fig. 6.9. The "true" electron density is calculated from the atomic coordinates. Only the DM algorithm was able to converge to the correct solution. The artifacts in the DM reconstruction are from the 5-fold NCS



(a) (b)

Figure 6.8 Error metrics vs Resolution. The labels indicate the half height of the Gaussian windowing, with 0Å indicating no Gaussian windowing. (a) R-factor (b) Image Correlation Coefficient.

being applied to the interface between the two virus particles in each unit cell. The refined support after the reconstruction at 5Å for the DM algorithm is shown in Fig. 6.9(f), and can be seen to be quite good.

A central slice of the DM algorithm reconstruction with no Gaussian weighting (at the final resolution extension) is shown in Fig. 6.10, and can be seen to be a reasonably good reconstruction. A contour plot of one of the lobes of the virus is shown in Fig. 6.11. The internal structure of the capsid is reconstructed reasonably well, with most of the main features correct.

In conclusion, the DM algorithm is able to successfully reconstruct the virus *ab initio* whereas conventional electron density modification failed. The reconstruction is of acceptable quality, and structures in the virus capsid can be clearly distinguished.

6.5 Reconstruction of a protein molecule

Icosahedral viruses present a rather ideal situation for *ab initio* phasing. The order of NCS is usually rather high, the position and orientation of the NCS axes are usually fixed or may have only one degree of freedom, and a good estimate of the envelope is usually at hand. These conditions are often not present with general proteins.

In the general case, the following parameters need to be known in order to directly apply IPAs to *ab initio* phasing.

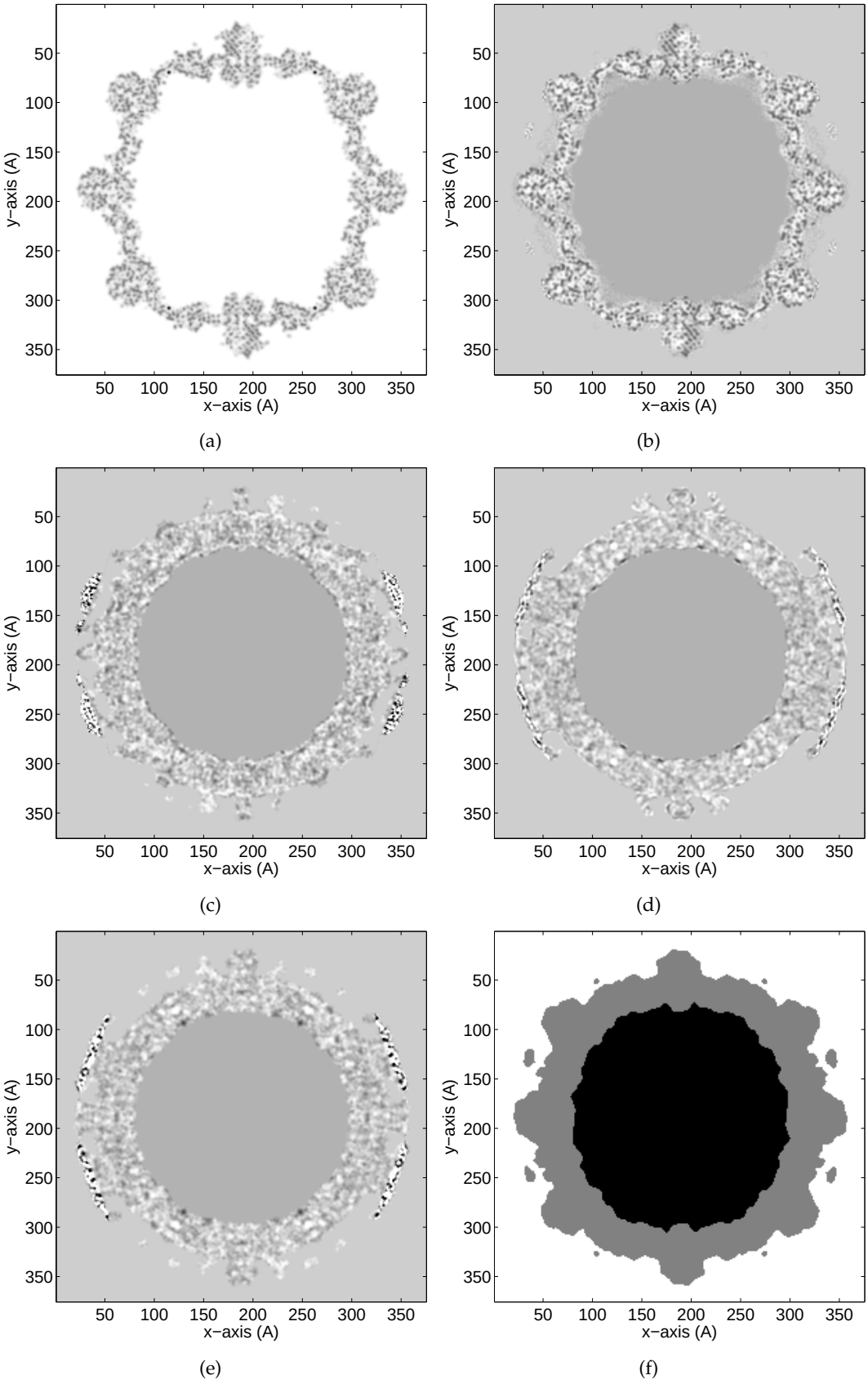
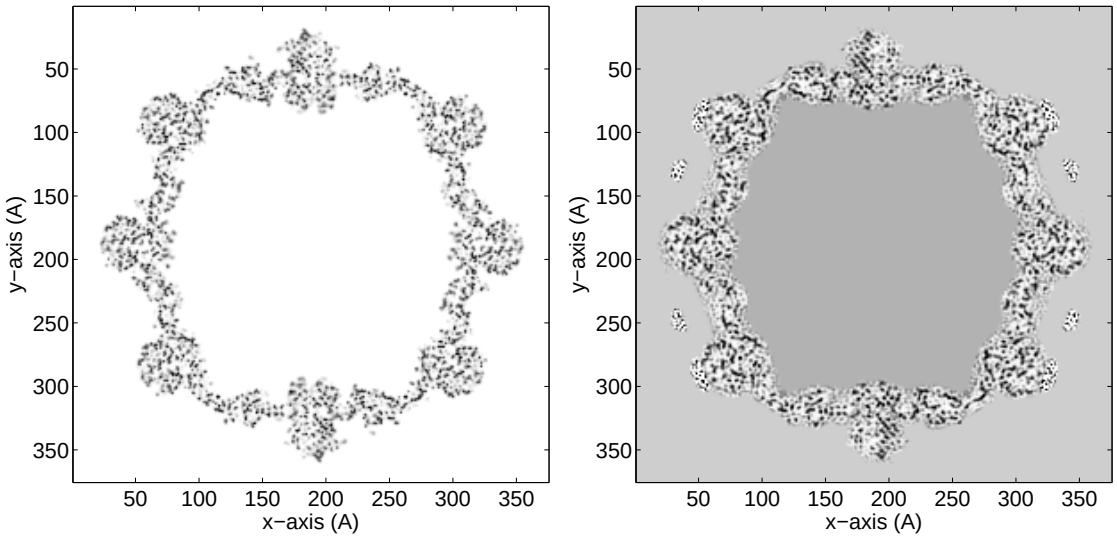
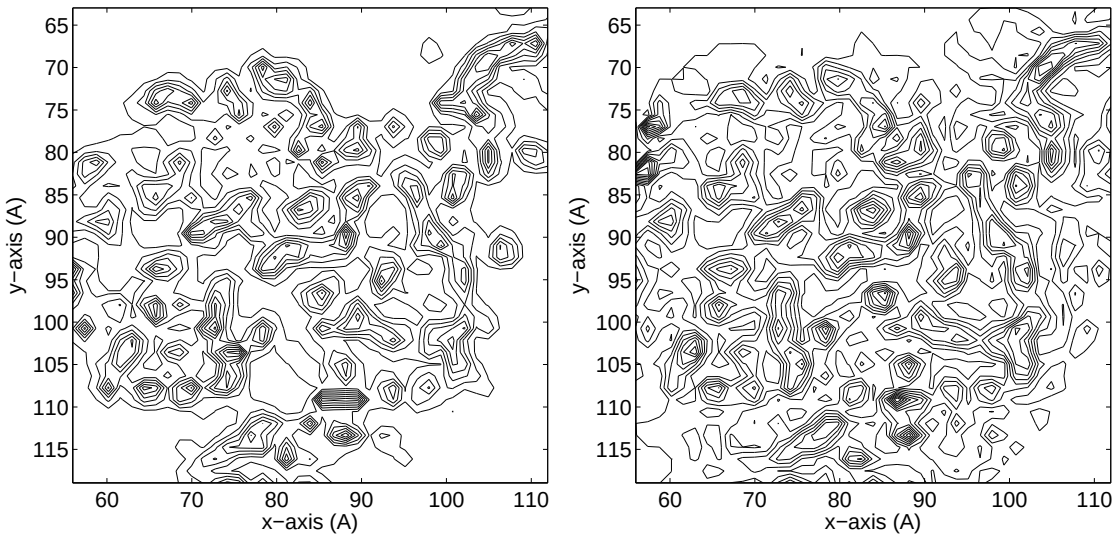


Figure 6.9 Cross sections through the centre of the virus unit cell at 5Å resolution extension (a) True electron density, (b) DM reconstruction, (c) GHIOER reconstruction, (d) GHIO reconstruction, (e) ER reconstruction (f) Estimated support at 5Å. The true electron density is found from the published structure, so the solvent levels are all zero. The virus reconstructions are from the experimental data, i.e. the actual virus crystal, where there are effectively two different solvent levels.



(a) (b)

Figure 6.10 Cross sections through the centre of the virus unit cell with no Gaussian windowing (2.8Å resolution) (a) True electron density, (b) DM reconstruction.



(a) (b)

Figure 6.11 Small sections through the centre of the virus unit cell with no Gaussian windowing (2.8Å resolution) (a) True electron density (b) DM Reconstruction.

1. The point group of the NCS.
2. The orientation of the NCS axes.
3. The position of the NCS axes.
4. The molecular (NCS) envelope.

In many cases the NCS symmetry and the orientation of the NCS axes can be determined from the Patterson function (rotation function). The remaining difficulties are determination of the position of the NCS axes (the translation problem) and an estimate (low resolution) of the molecular envelope.

In some cases a low resolution estimate of the molecular envelope may be available from, for example, solution scattering or electron microscopy. Alternatively, an estimate of the molecular envelope may be obtained using crystal solvent contrast data as described in Ch. 4. Solution of the translation problem *ab initio* is in practice extremely difficult.

Not all of these difficulties are addressed here, but for the purpose of evaluating the algorithms fairly minimal additional information is assumed. It is assumed that the degree and orientation of the NCS axes has been obtained from the Patterson function using the rotation function and that a low resolution (10Å) estimate of the molecular envelope has been obtained (e.g. from solution scattering, electron microscopy, contrast variation, or some other technique).

6.5.1 Methods

Experimental diffraction data from the tryptophanase protein from *proteus vulgaris* [55] (PDB:1ax4), was used. The protein has $P2_12_12_1$ crystallographic symmetry, giving an order 4 crystallographic symmetry. Additionally, each asymmetric unit has two orthogonal non-crystallographic two-fold rotation axes, giving a 4-fold NCS. The two orthogonal non-crystallographic axes imply a third orthogonal two-fold rotation axis. The unit cell dimensions are $115 \times 118 \times 154$Å. A diagram of the protein and its symmetry axes is shown in Fig. 6.13, where each oligomer is coloured in a different colour. Experimental diffraction data between approximately 12 and 2Å resolution are available, although a large number of the lower order data and some higher order data are also missing as shown in Fig. 6.12, which shows a central slice of the available Fourier data. Using the published structure, standard crystallographic programs [72] were used to calculate the electron density on a $110 \times 114 \times 146$ grid, corresponding to a grid spacing of 1Å and a resolution of 2Å.

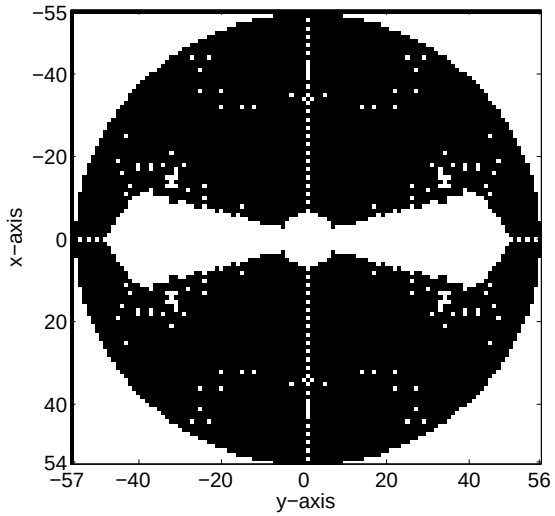


Figure 6.12 A central slice of the Fourier magnitudes showing the missing data in white.

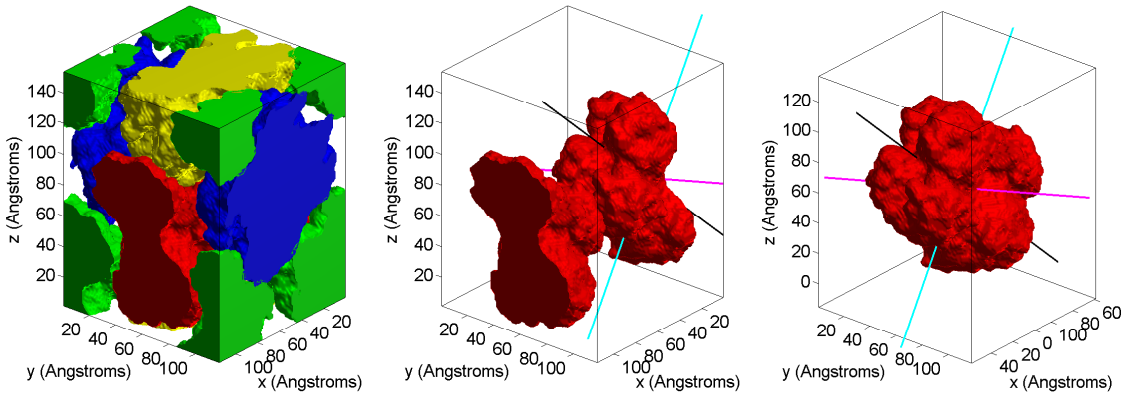


Figure 6.13 The tryptophanase protein from *proteus vulgaris* in a $P2_12_12_1$ unit cell. (a) Full protein in the unit cell, with each oligomer in a different colour, (b) the red oligomer and the NCS rotation axes, (c) the red oligomer with axes shifted to the centre of the unit cell.

6.5.1.1 Determining envelope position

It is assumed that the orientations of the NCS axes are known along with a low resolution estimate of the oligmer envelope. For simulation purposes, the envelope was obtained by low-pass filtering the electron density with a Gaussian with a width at half-height of 10Å and then thresholding the result to create the envelope shown in Fig. 6.13(b). The positions and orientations of the NCS axes in the envelope are known (if not they can usually be found by inspection of the envelope), but there are six possible orientations of the 4-fold NCS envelope in the unit cell that are consistent with the NCS in the crystal. The correct orientation of the NCS axes and the position of the NCS origin in the crystal were determined as follows.

For each of the six possible orientations of the envelope, the envelope was shifted to all grid points in the asymmetric unit, all the other envelopes in the unit cell were generated by the crystallographic symmetry, and the number of overlapping grid points counted.

Define the 2-fold symmetry three axes of the oligomer as m_{olig}, b_{olig} and c_{olig}, and the three axes of the unit cell as m_{crys}, b_{crys} and c_{cryst}, where the variable name corresponds to the first letter of the colour of the axes in Fig. 6.13. Plots of the overlap in the x-y plane through the minimum for each of the 6 orientations are shown in Fig. 6.14. The other minima are a result of the crystallographic symmetry. The value of the minimum overlap in pixels for each orientation out of $110 \times 114 \times 146 = 1830840$ pixels is listed in Table 6.1. There are clearly two possible solutions (orientations 1 and 3) corresponding to two possible orientations of the envelope. Orientation 1 is correct and a small overlap is obtained when the cyan and black NCS axes are interchanged. This is a result of the magenta axis being close to parallel ($13.8°$) to the unit cell y-axis, so that when the black and cyan axes are interchanged, the similar unit cell dimensions of the x and z axes also results in a minimal amount of overlap. Orientation 1 was used in the structure determination.

6.5.1.2 Resolution Extension

The low resolution envelope is used for the support and NCS constraint at the lowest resolution. The envelope is refined at the beginning of each resolution extension in a similar way to that described in Sec. 6.4.1 except that there is only one solvent level - that outside the molecule. Gaussian weighting of the diffraction data and application of the NCS averaging are the same as described for the virus in Sec. 6.4.1.

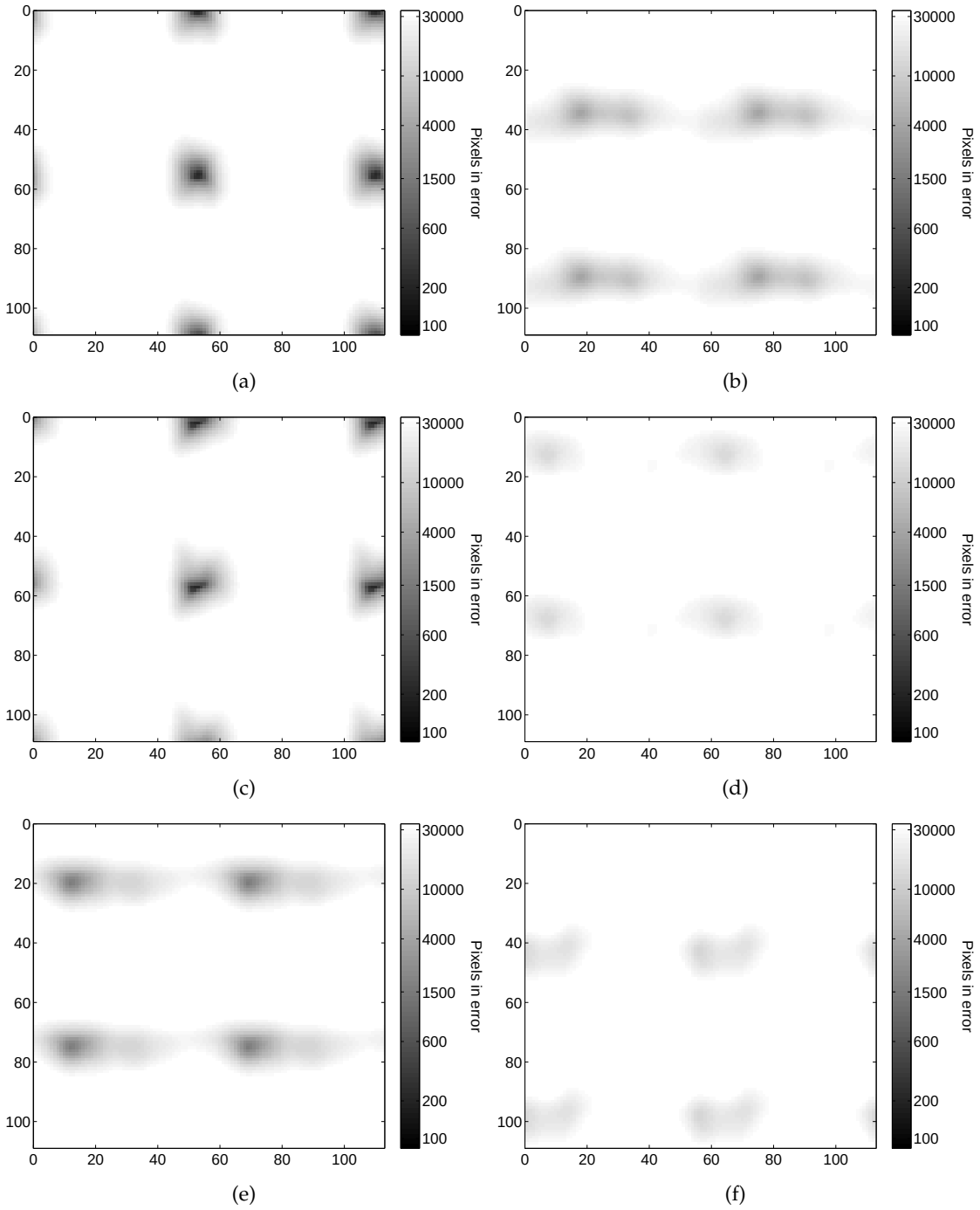


Figure 6.14 Steric search plots. (a) Orientation 1 (b) Orientation 2 (c) Orientation 3 (d) Orientation 4 (e) Orientation 5 (f) Orientation 6.

Table 6.1 Results of the steric search.

Orientation	m_{cryst}	b_{cryst}	c_{cryst}	Overlap (grid points)
1	m_{olig}	b_{olig}	c_{olig}	4
2	b_{olig}	m_{olig}	c_{olig}	1440
3	m_{olig}	c_{olig}	b_{olig}	28
4	c_{olig}	b_{olig}	m_{olig}	11924
5	c_{olig}	m_{olig}	b_{olig}	1312
6	b_{olig}	c_{olig}	m_{olig}	12080

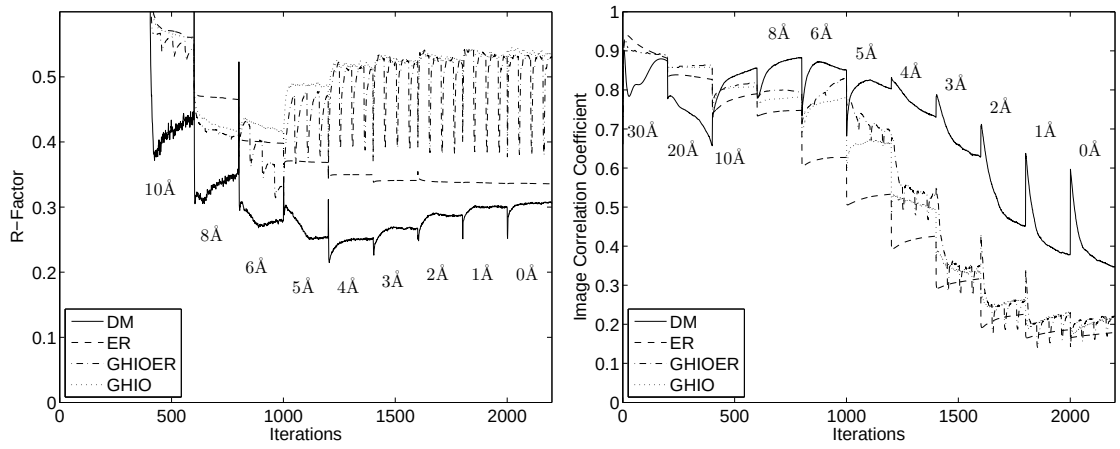
6.5.2 Reconstruction

The ER, GHIO, GHIOER and DM with $\beta = 0.7$ algorithms were used. The electron density outside the support, ρ_s, was set to 0.1. The algorithm was run with resolution extensions at 30, 20, 10, 8, 6, 5, 4, 3, 2, 1 Å and 200 cycles at each resolution using a $110 \times 114 \times 146$ grid, corresponding to a grid spacing of 1Å. At the final resolution extension, no Gaussian windowing was used. Progress of the reconstruction was monitored by calculating the R-factor and image correlation coefficient as defined in Sec. 6.4.2. These quantities were also calculated as a function of resolution at the end of each resolution extension.

The R-factor and image correlation coefficient versus iteration are shown in Fig. 6.15. The DM algorithm performed the best, with the other three algorithms unable to make much progress past 6Å. Of the three failed algorithms, the ER performed worse than the GHIO and GHIOER algorithms.

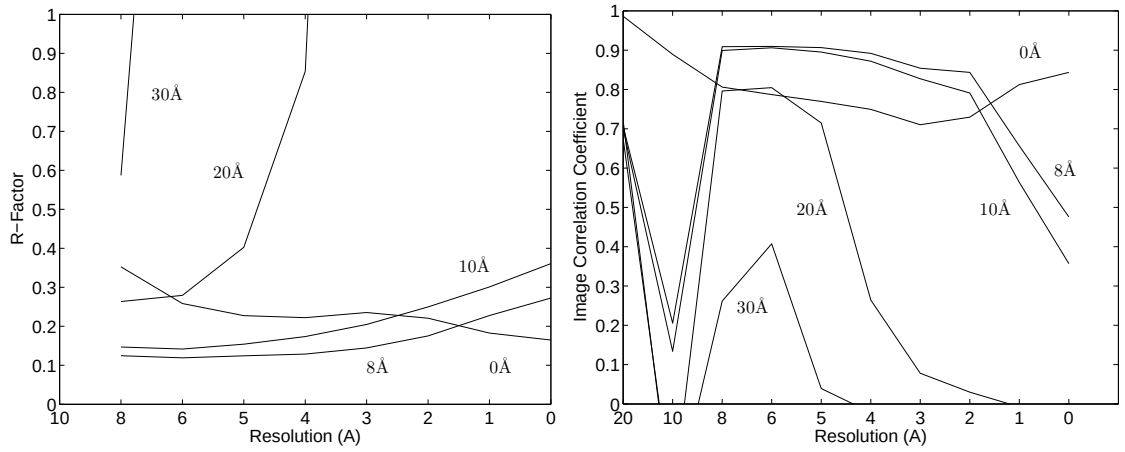
The error metrics versus resolution at different resolution extensions are shown in Fig. 6.16 for the DM algorithm. As the bootstrapping proceeds, the improvement of the metrics at the higher resolutions can clearly be seen. The poor error metrics at low resolution (10Å) is likely to be a result of the missing low resolution Fourier magnitudes.

Central slices of the electron density through the unit cell with no Gaussian windowing are shown for each algorithm in Fig. 6.17. Only the DM algorithm was able to converge to the correct solution. The DM algorithm contour maps of a small section are shown in Fig. 6.18. The reconstruction is acceptable and the main features can be seen, but the reconstruction is not as good as the reconstruction for the virus. Note, however, that since the "true" electron density is calculated from the atomic coordinates, even though it is on the same grid spacing as the reconstructed electron density, it will have an inherently higher resolution. A 3-D plot



(a) (b)

Figure 6.15 Error metrics vs Iteration. The half-height of the gaussian windowing for each resolution is shown, with 0Å indicating no Gaussian windowing. (a) R-factor vs Iteration (b) Image Correlation Coefficient vs Iteration.



(a) (b)

Figure 6.16 Error metrics vs Resolution. The labels indicate the half height of the Gaussian windowing, with 0Å indicating no Gaussian windowing. 0Å indicates no Gaussian windowing. (a) R-factor vs Resolution (b) Image Correlation Coefficient vs Resolution.

of a small section of the protein that shows the reconstruction superimposed upon the true atomic structure is shown Fig. 6.19 (provided by Richard Kingston).

As with the virus example, the number of iterations can be reduced by extending the resolution once the error metric increases. The improvement obtained is shown in Fig. 6.20, and the number of iterations is reduced to about half.

Reconstruction of a protein is a more difficult problem than reconstruction of a virus due to the lower order NCS and the difficulty of estimating the NCS parameters. In this particular case, the large amount of unmeasured low resolution Fourier magnitudes makes the situation worse. Nevertheless, the protein was successfully reconstructed *ab initio* to around 3Å resolution.

6.6 Conclusions

NCS is a powerful constraint for *ab initio* phasing of macromolecular crystals. It is shown that 3-fold or higher symmetry should alone be sufficient to reconstruct the electron density. Iterative projection algorithms are developed incorporating NCS and are shown to be effective.

The high order symmetry in a virus particle allows the NCS symmetry parameters to be found easily. The *ab initio* reconstruction of a virus molecule to high resolution was successfully demonstrated. More difficulty was encountered with the reconstruction of a protein, where the estimation of the symmetry parameters, especially the position of the NCS axes, was more difficult. Assuming that the support of each oligomer is known, a steric search was shown to be successful in estimating the position of the oligomers and the NCS origin. The protein reconstruction structure was also determined *ab initio*, and although the reconstruction is less accurate than the virus reconstruction, but a good reconstruction was nonetheless achieved.

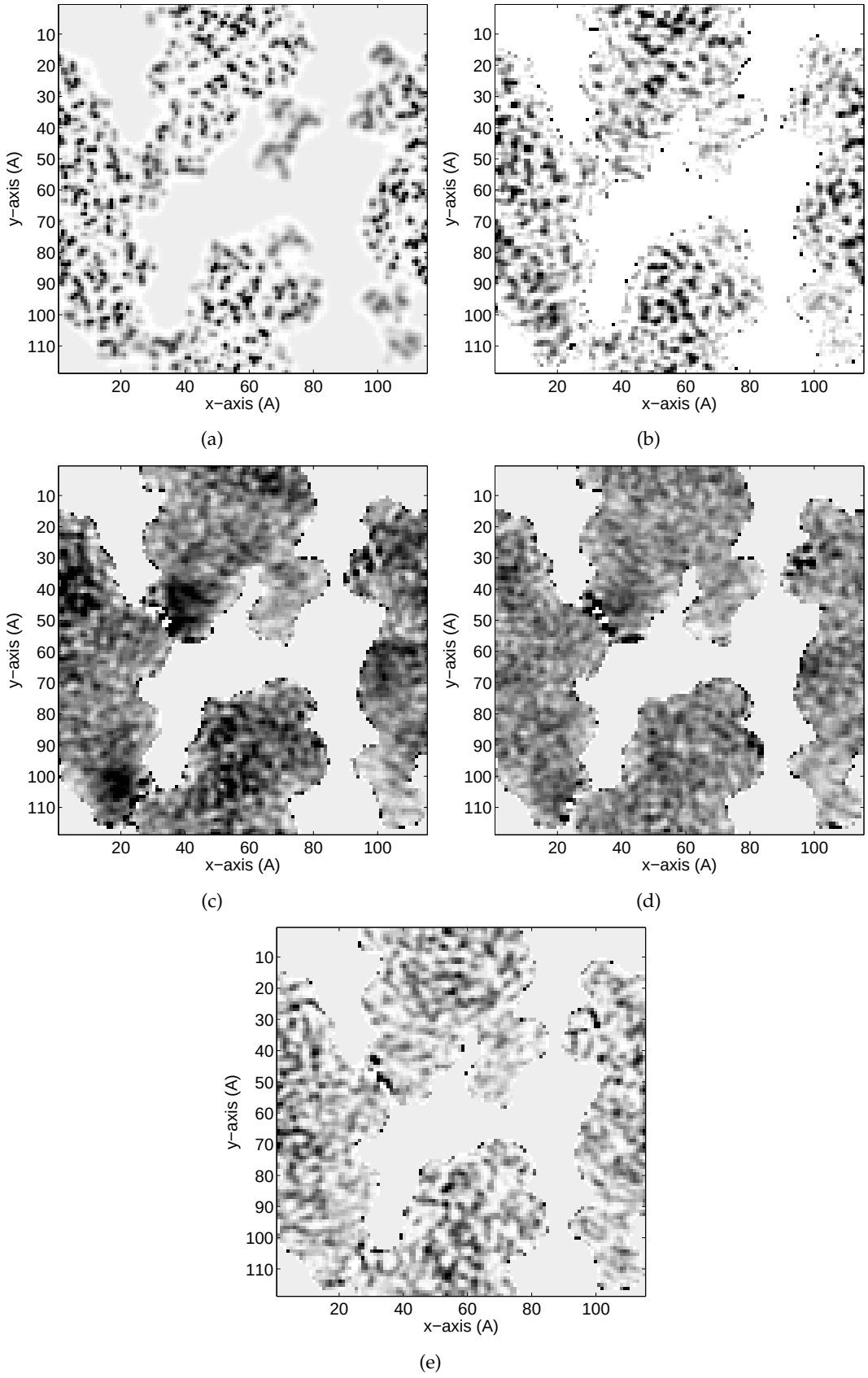


Figure 6.17 Cross sections through the centre of the unit cell at the final resolution extension with no Gaussian windowing. (a) True electron density, (b) DM reconstruction, (c) ER reconstruction, (d) GHIOER reconstruction, (e) GHIO reconstruction.

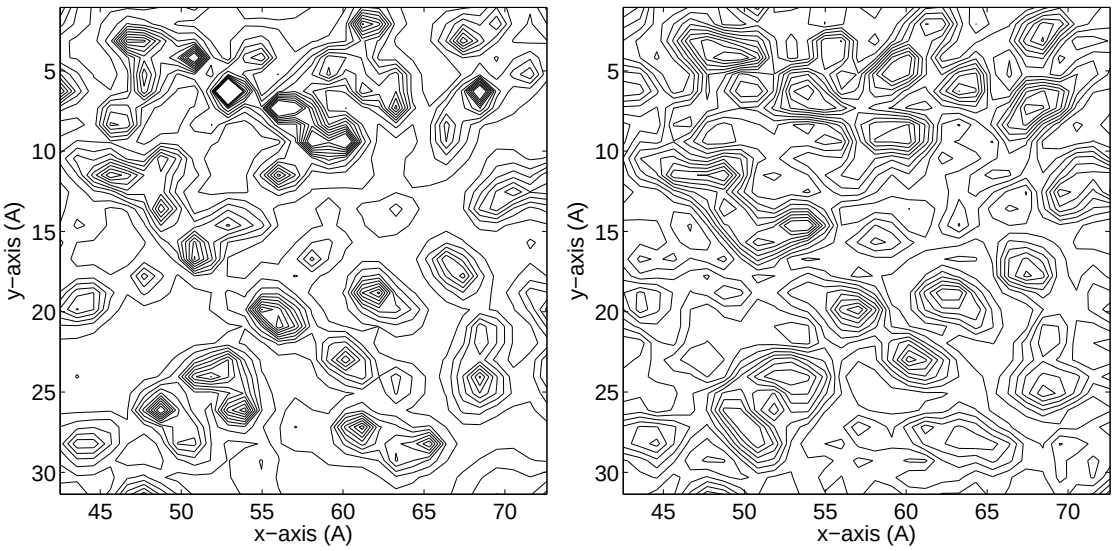


Figure 6.18 Small sections through the centre of the unit cell at the final resolution extension with no Gaussian windowing. (a) True electron density (b) DM Reconstruction.

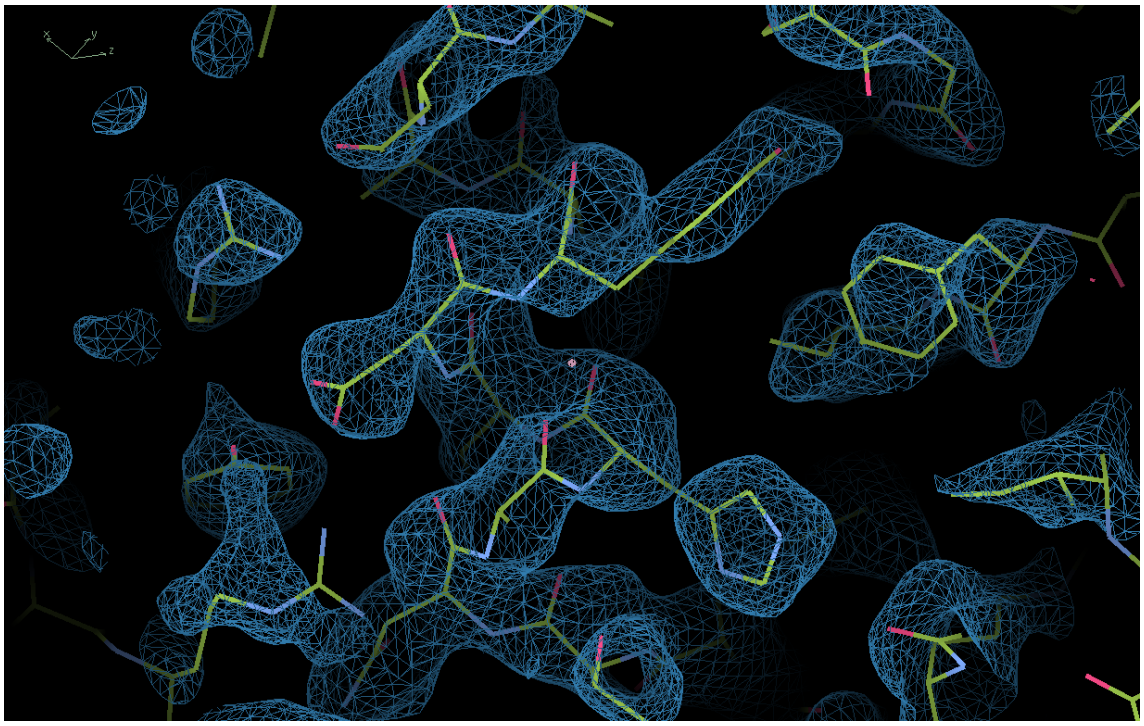


Figure 6.19 3D reconstruction of a small section of the protein showing the correlation with the atomic model. With thanks to Richard Kingston.

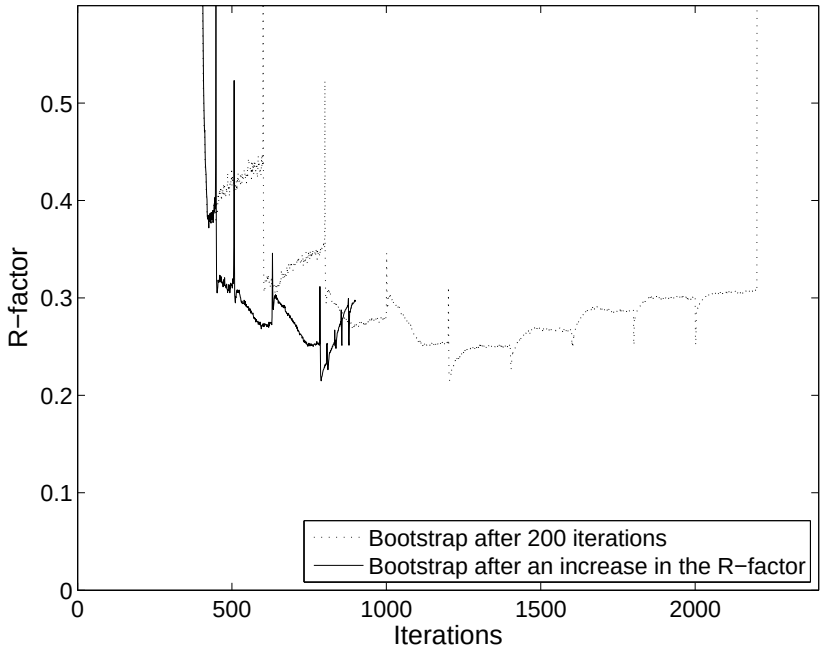


Figure 6.20 R-factor vs Resolution with bootstrapping when an increase in the R-factor is detected.

Chapter 7

Conclusions

Iterative projection algorithms are a powerful and relatively easy-to-use method for solving a large class of inverse problems, and have been used in many branches of science and mathematics. Recently, more sophisticated algorithms have been developed, and along with increases in computing power, have allowed larger and more difficult problems to be solved.

The study of IPAs in mathematics has focussed mainly on convex problems, with less work done in the considerably more difficult case of non-convex constraints. Chapter 2 introduced a model for the progress towards the solution of IPAs, where the progress of IPAs are divided into the initial, search, and final convergence phases. Convergence is modeled as a constant probability process, which experiments seem to confirm. Problems with discrete constraints have a small radius of convergence, and thus spend more time in the search phase. Problems with continuous constraints may spend more time in the convergence phase if they have a small "effective angle" between the constraint sets. A solution has been proposed, which is to use some CP recursions, which are designed to converge quickly when there are small angle problems.

IPAs have been used in crystallography for phase retrieval, but generally only the ER or RP (solvent flipping) algorithms are used. These simple algorithms have poor global convergence properties and stagnate easily, requiring experimental phases to be found so that the algorithm can be started close to the solution. The use of more sophisticated IPAs should allow a reduction of experimental work, replacing costly laboratory work with cheaper computing time, as well as raising the possibility of *ab initio* phasing.

Determining the molecular envelope is an important precursor to attempts at *ab initio* phasing. This problem corresponds to reconstructing a binary image from undersampled Fourier amplitudes. It is shown that with appropriate constraints

this problem is likely to have a unique solution and an effective IPA has been developed. This approach is shown to be feasible for reconstructing molecular envelopes from solvent contrast data.

Symmetry provides redundancy in images and occurs frequently in macromolecules. Methods for incorporating symmetry constraints in IPAs have been described and difficulties due to the necessity to interpolate sampled images discussed. Symmetry constraints are used in an *ab initio* phasing scheme for macromolecular crystallography using IPAs. Application to an icosahedral virus and symmetric protein indicates that this approach has considerable potential for *ab initio* phasing in macromolecular crystallography

The work completed here suggests a number of fruitful avenues for future research.

1. The behavior of IPAs with non-convex constraints is complex and is still poorly understood. Further analytic and numerical studies are needed to increase our understanding of these algorithms which should lead to more and better applications.
2. The fusion of probabilistic methods and IPAs, which combines the noise-stability and performance of probabilistic methods with the speed and search capabilities of IPAs has considerable potential. This can be currently be done by using an IPA to get near the vicinity of the solution and then using probabilistic methods to fine tune the solution, but a combined method would be better. Some work has been done [29], and applications to crystallography may prove fruitful.
3. The development of methods to find the NCS symmetry parameters as part of the algorithm would prove very valuable for *ab initio* phasing. A method has been suggested, but a considerable amount of work is needed.
4. Verification of the solvent contrast methods with experimental data would allow the true potential of these methods to be assessed.
5. Finally, computing resources can still limit high resolution reconstruction of very large macromolecular assemblies. Formulating these algorithms to utilize parallel and other high speed computer architectures will extend the power of these methods.

Bibliography

[1] A. Ambardar, *Analog and digital signal processing*. Brooks/Cole, 1999.

[2] J. Drenth, *Principles of X-ray Crystallography*. Springer-Verlag, 1994.

[3] R. P. Millane, "Phase retrieval in crystallography and optics," *J. Opt. Soc. Am. A*, vol. 7, pp. 394–411, 1990.

[4] H. N. Chapman, A. Barty, M. J. Bogan, S. Boutet, M. Frank, S. P. Hau-Riege, S. Marchesini, B. W. Woods, S. Bajt, W. H. Benner, R. A. London, E. Plnjes, M. Kuhlmann, R. Treusch, S. Dsterer, T. Tschentscher, J. R. Schneider, E. Spiller, T. Mller, C. Bostedt, M. Hoener, D. A. Shapiro, K. O. Hodgson, D. van der Spoel, F. Burmeister, M. Bergh, C. Caleman, G. Huldt, M. M. Seibert, F. R. N. C. M, R. W. Lee, A. Szke, N. Timneanu, and J. Hajdu, "Femtosecond diffractive imaging with a soft-x-ray free-electron laser," *Nature Physics*, vol. 2, pp. 839–843, 2006.

[5] P. Thibault, V. Elser, C. Jacobsen, D. Shapiro, and D. Sayre, "Reconstruction of a yeast cell from x-ray diffraction data," *Acta Cryst. A*, vol. 62, pp. 248–261, 2006.

[6] J. C. H. Spence and B. Doak, "Single molecule diffraction," *Phys. Review Letters*, vol. 92, p. 198102, 2004.

[7] Commission on International Tables, *International Tables for Crystallography*. International Union of Crystallography, 2010.

[8] G. Bricogne, "Direct phase determination by entropy maximization and likelihood ranking: status report and perspectives," *Acta Cryst. D*, vol. 49, pp. 37–60, 1993.

[9] W. A. Hendrickson, J. L. Smith, R. P. Phizackerley, and E. A. Merritt, "Crystallographic structure analysis of lamprey hemoglobin from anomalous dispersion of synchrotron radiation," *Proteins*, vol. 4, pp. 77–88, 1988.

[10] M. G. Rossmann and D. M. Blow, "The detection of sub-units within the crystallographic asymmetric unit." *Acta Cryst.*, vol. 15, pp. 24–31, 1962.

[11] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of the phase from image and diffraction plane pictures," *Optik*, vol. 35, p. 237, 1972.

[12] J. P. Abrahams and A. G. W. Leslie, "Methods used in the structure determination of bovine mitochondrial F_1 ATPase," *Acta Cryst. D*, vol. 52, pp. 30–42, 1996.

[13] G. Oszlányi and A. Sütö, "Ab initio structure solution by charge flipping," *Acta Cryst. A*, vol. 60, p. 134, 2004.

[14] J. Douglas and H. H. Rachford, "On the numerical solution of the heat conduction problem in 2 and 3 space variables," *Trans. Amer. Math. Soc.*, vol. 82, pp. 421–439, 1956.

[15] J. R. Fienup, "Phase retrieval algorithms: a comparison," *Applied Optics*, vol. 21(15), pp. 2758–2769, 1982.

[16] V. Elser, "Phase retrieval by iterated projections," *J. Opt. Soc. Am. A*, vol. 20, pp. 40–55, 2003.

[17] H. H. Bauschke, P. L. Combettes, and D. R. Luke, "Hybrid projection–reflection method for phase retrieval," *J. Opt. Soc. Am. A*, vol. 20, no. 6, pp. 1025–1034, June 2003.

[18] R. P. Millane and W. J. Stroud, "Reconstructing symmetric images from their undersampled Fourier intensities," *J. Opt. Soc. Am. A*, vol. 14, pp. 568–579, 1997.

[19] D. R. Luke, "Relaxed averaged alternating reflections for diffraction imaging," *Inverse Problems*, vol. 21, pp. 37–50, 2005.

[20] D. Sayre, "The squaring method: a new method for phase determination," *Acta Cryst*, vol. 5, pp. 60–65, 1952.

[21] T. Debaerdemaeker, C. Tate, and M. M. Woolfson, "On the application of phase relationships to complex structures. xxvi. developments of the Sayre-equation tangent formula," *Acta Cryst. A*, vol. 44, pp. 353–357, 1988.

[22] A. T. Brünger, "Free R value: a novel statistical quantity for assessing the accuracy of crystal structures," *Nature*, vol. 355(6359), pp. 472–475, 1992.

[23] ——, "Assessment of phase accuracy by cross validation: The free R value. methods and applications," *Acta Cryst. D*, vol. 49, pp. 24–36, 1993.

[24] V. Elser, I. Rankenburg, and P. Thibault, "Searching with iterated maps," *Proc. Nat. Acad. Sci.*, vol. 104, pp. 418–423, 2007.

[25] J. Milnor, "On the concept of attractor," *Communications of Mathematical Physics*, vol. 99, pp. 177–195, 1985.

[26] V. Elser, "Solution of the crystallographic phase problem by iterated projections," *Acta Cryst. A*, vol. 59, pp. 201–209, 2003.

[27] J. R. Fienup and C. C. Wackerman, "Phase-retrieval stagnation problems and solutions," *J. Opt. Soc. Am. A*, vol. 3(11), pp. 1897–1907, 1986.

[28] S. Gravel and V. Elser, "Divide and concur: A general approach to constraint satisfaction," *Phys. Rev. E*, vol. 78, no. 3, p. 036706, 2008.

[29] J. S. Yedida, Y. Wang, and S. C. Draper, "Divide & concur and difference-map bp decoders for ldpc codes," *arXiv*, vol. arXiv:1001.1730v1, 2010.

[30] S. Eisebitt, J. Lüning, W. F. Schlotter, M. Lörgen, O. Hellwig, W. Eberhardt, and J. Stöhr, "Lensless imaging of magnetic nanostructures by x-ray spectroholography," *Nature*, vol. 432, pp. 885–888, 2004.

[31] D. I. Svergun and H. B. Stuhrmann, "New developments in direct shape determination from small angle scattering 1. theory and model calculations," *Acta. Cryst. A*, vol. 47, pp. 736–744, 1991.

[32] P. Chacon, J. F. Diaz, F. Moran, and J. M. Andreu, "Reconstruction of protein form with x-ray solution scattering and a genetic algorithm," *J. Mol. Biol.*, vol. 299, pp. 1289–1302, 2000.

[33] D. I. Svergun, M. V. Petoukhov, and M. H. J. Koch, "Determination of domain structure of proteins from x-ray solution scattering," *Biophysical J.*, vol. 80, pp. 2946–2953, 2001.

[34] Q. Hao, "Macromolecular envelope determination and envelope-based phasing," *Acta Cryst.*, vol. D62, pp. 909–914, 2006.

[35] D. I. Svergun, "Small-angle scattering studies of macromolecular solutions," *J. Appl. Cryst.*, vol. 40, pp. s10–s17, 2007.

[36] C. D. Putnam, "X-ray solution scattering (SAXS) combined with crystallography and computation: defining accurate macromolecular structures, conformations and assemblies in solution," *Quarterly Reviews of Biophysics*, vol. 40, pp. 191–285, 2007.

[37] H. B. Stuhrmann, "Small-angle scattering and its interplay with crystallography, contrast variation in SAXS and SANS," *Acta. Cryst. A*, vol. 64, pp. 181–191, 2008.

[38] E. J. Dodson, "Using electron-microscopy images as a model for molecular replacement," *Acta Cryst.*, vol. D57, pp. 1405–1409, 2001.

[39] J. Navaza, "Combining x-ray and electron-microscopy data to solve crystal structures," *Acta Cryst. D*, vol. 64, pp. 70–75, 2008.

[40] Y. Xiong, "From electron microscopy to x-ray crystallography: molecular-replacement case studies," *Acta Cryst.*, vol. D64, pp. 76–82, 2008.

[41] W. L. Bragg and M. F. Perutz, "The external form of the haemoglobin molecule, i," *Acta Cryst.*, vol. 5, pp. 277–283, 1952.

[42] C. W. Carter, K. V. Crumley, D. E. Coleman, F. Hage, and G. Bricogne, "Direct phase determination for the molecular envelope of Tryptophanyl-tRNA synthetase from *Bacillus stearothermophilus* by x-ray contrast variation," *Acta Cryst. A*, vol. 46, pp. 57–68, 1990.

[43] R. Fourme, W. Shepard, R. Kahn, G. l'Hermite, and I. L. de La Sierra, "The multiwavelength anomalous solvent contrast (MASC) method in macromolecular crystallography," *J. Synchrotron Rad.*, vol. 2, pp. 36–48, 1995.

[44] W. Shepard, R. Kahn, M. Ramin, and R. Fourme, "Low-resolution phase information in multiple-wavelength anomalous solvent contrast variation experiments," *Acta Cryst. D*, vol. 56, pp. 1288–1303, 2000.

[45] J. Badger, "Determination of protein and solvent volumes in protein crystals from contrast variation data," *Basic Life Sci.*, vol. 64, pp. 333–343, 1996.

[46] W. A. Hendrickson, "Determination of macromolecular structures from anomalous diffraction of synchrotron radiation," *Science*, vol. 254, pp. 51–58, 1991.

[47] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalovand, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, 2000.

[48] H. Miyatake, Y. Hata, T. Fujii, K. Hamada, K. Morihara, and Y. Katsube, "Crystal-structure of the unliganded alkaline protease from *Pseudomonas aeruginosa* IFO3080 and its conformational-changes on ligand binding," *J. Biochemistry*, vol. 118 (3), pp. 474–479, 1995.

[49] D. D. Leonidas, E. H. Vatzaki, H. Vorum, J. E. Celis, P. Madsen, and K. R. Acharya, "Structural basis for the recognition of carbohydrates by human galectin-7," *Biochemistry*, vol. 37 (40), pp. 13 930–13 940, 1998.

[50] B. C. Wang, "Resolution of phase ambiguity in macromolecular crystallography," *Methods in Enzymology*, vol. 115, pp. 90–112 Part B, 1985.

[51] A. G. W. Leslie, "A reciprocal-space method for calculating a molecular envelope using the algorithm of B.C. Wang," *Acta Cryst. A*, vol. 43, pp. 134–136, 1987.

[52] K. Cowtan, "DM: An automated procedure for phase improvement by density modification," *Joint CCP4 and ESF-EACBM Newsletter on Protein Crystallography*, vol. 31, pp. 34–38, 1994.

[53] G. J. Kleywegt, "Use of non-crystallographic symmetry in protein structure refinement," *Acta Cryst.*, vol. D, pp. 842–857, 1996.

[54] M. G. Rossmann and D. M. Blow, "Determination of phases by the conditions of non-crystallographic symmetry." *Acta Cryst.*, vol. 16, pp. 39–45, 1963.

[55] M. N. Isupov, A. A. Antson, E. J. Dodson, G. G. Dodson, I. S. Dementieva, L. N. Zakomirdina, K. S. Wilson, Z. Dauter, A. A. Levedev, and E. H. Harutyunyan, "Crystal structure of tryptophanase," *J. Mol. Biol.*, vol. 276, pp. 603–623, 1998.

[56] R. A. Crowther and D. M. Blow, "A method of positioning a known molecule in an unknown crystal structure," *Acta Cryst.*, vol. 23, pp. 544–548, 1967.

[57] A. G. Urzhumtsev, E. A. Vernoslova, and A. D. Podjarny, "Approaches to very low resolution phasing of the ribosome 50S particle from *Thermus thermophilus* by the few-atoms-models and molecular-replacement methods," *Acta Cryst. D*, vol. 52, pp. 1092–1097, 1996.

[58] P. Main and M. G. Rossmann, "Relationships among structure factors due to identical molecules in different crystallographic environments," *Acta Cryst.*, vol. 21, pp. 67–72, 1966.

[59] R. A. Crowther, "The use of non-crystallographic symmetry for phase determination," *Acta Cryst.*, vol. B25, pp. 2571–2580, 1969.

[60] G. Bricogne, "Geometric sources of redundancy in intensity data and their use for phase determination," *Acta Cryst.*, vol. A30, pp. 395–405, 1974.

[61] J. Tsao, M. Chapman, and M. Rossmann, "Ab initio phase determination for viruses with high symmetry: a feasibility study," *Acta Cryst. A*, vol. 48, pp. 293–301, 1992.

[62] J. Tsao, M. Chapman, M. Agbandje, W. Keller, K. Smith, H. Wu, M. Luo, T. Smith, M. Rossmann, and R. Compans et al, "The three-dimensional structure of canine parvovirus and its functional implications," *Science*, vol. 251(5000), pp. 1456–1464, 1991.

[63] S. Munshi, L. Liljas, and J. E. Johnson, "Structure determination of *Nudaurelia capensis* ω virus," *Acta Cryst. D*, vol. 54, pp. 1295–1305, 1998.

[64] H. Naitow, Y. Morimoto, H. Mizuno, H. Kano, T. Omura, M. Koizumi, and T. Tsukihara, "A low-resolution structure of rice dwarf virus determined by ab initio phasing," *Acta Cryst. D*, vol. 55, pp. 77–84, 1999.

[65] J. Taka, H. Naitow, M. Yoshimura, N. Miyazaki, A. Nakagawa, and T. Tsukihara, "Ab initio crystal structure determination of spherical viruses that exhibit a centrosymmetric location in the unit cell," *Acta Cryst. D*, vol. 61, pp. 1099–1106, 2005.

[66] E. Arnold and M. G. Rossmann, "Effect of errors, redundancy, and solvent content in the molecular replacement procedure for the structure determination of biological macromolecules," *Proc. Natl. Acad. Sci. U.S.*, vol. 83, pp. 5489–5493, 1986.

[67] R. P. Millane, "Phase problems for periodic images: effects of support and symmetry," *J. Opt. Soc. Am. A*, vol. 10, pp. 1037–1045, 1993.

[68] V. Elser and R. P. Millane, "Reconstruction of an object from its symmetry-averaged diffraction pattern," *Acta Cryst. A*, vol. 64, pp. 273–279, 2008.

[69] N. J. Dimmock, A. J. Easton, and K. Leppard, *Introduction to Modern Virology sixth edition*. Blackwell Publishing, 2007.

[70] E. E. Fry, J. Grimes, and D. I. Stuart, "Virus crystallograpy," *J. Mol. Biol.*, vol. 12, pp. 13–s23, 1999.

[71] Y. Wada, H. Tanaka, E. Yamashita, C. Kubo, T. Ichiki-Uehara, E. Nakazono-Nagaoka, T. Omura, and T. Tsukihara, "The structure of melon necrotic spot virus determined at 2.8Å resolution," *Acta Cryst. F*, vol. 64, pp. 8–13, 2008.

[72] Collaborative Computational Project, Number 4. 1994, "The CCP4 suite: Programs for protein crystallography," *Acta Cryst. D*, vol. 50, pp. 760–763, 1994.